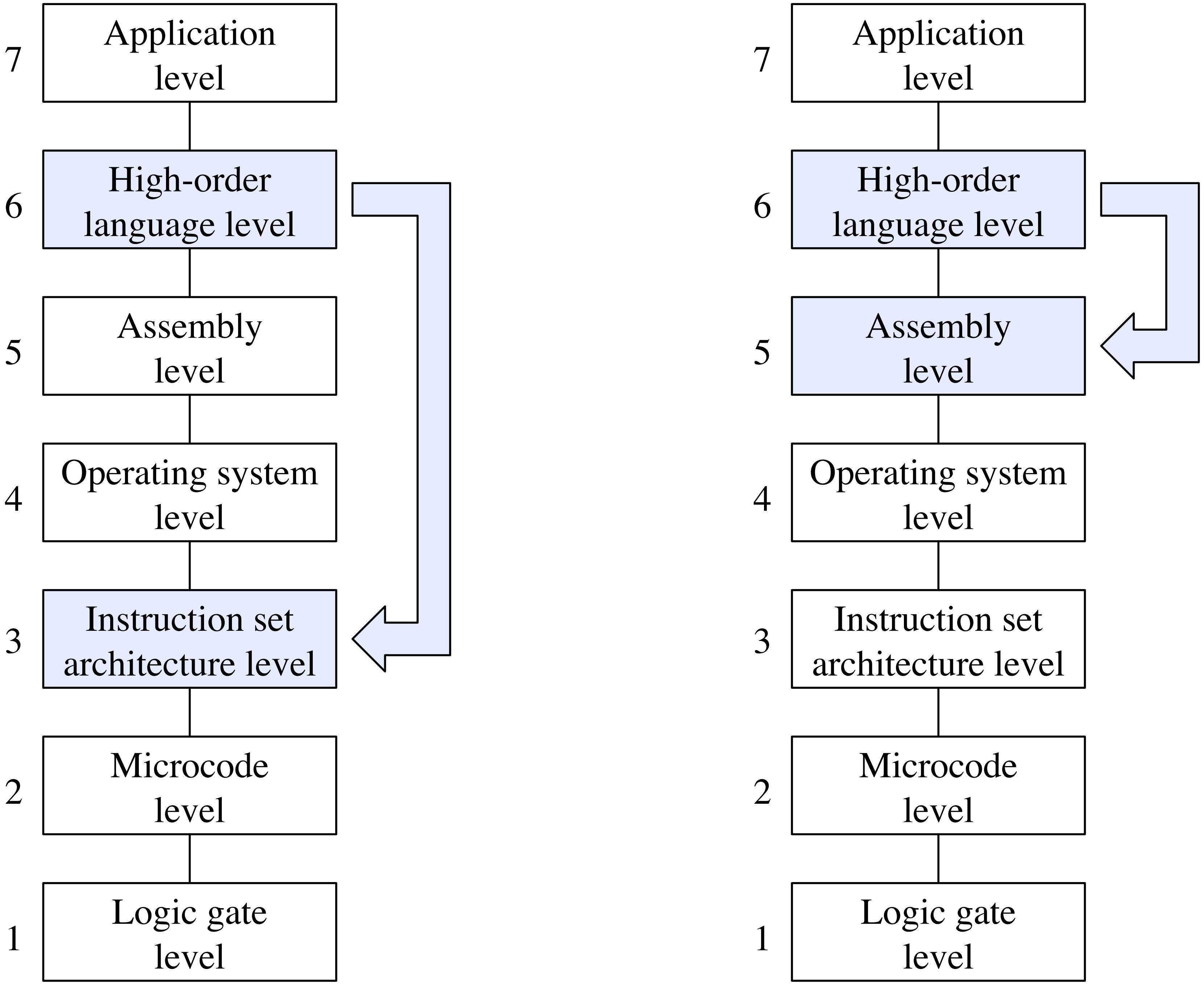
Computer Systems

Sixth edition

Chapter 6

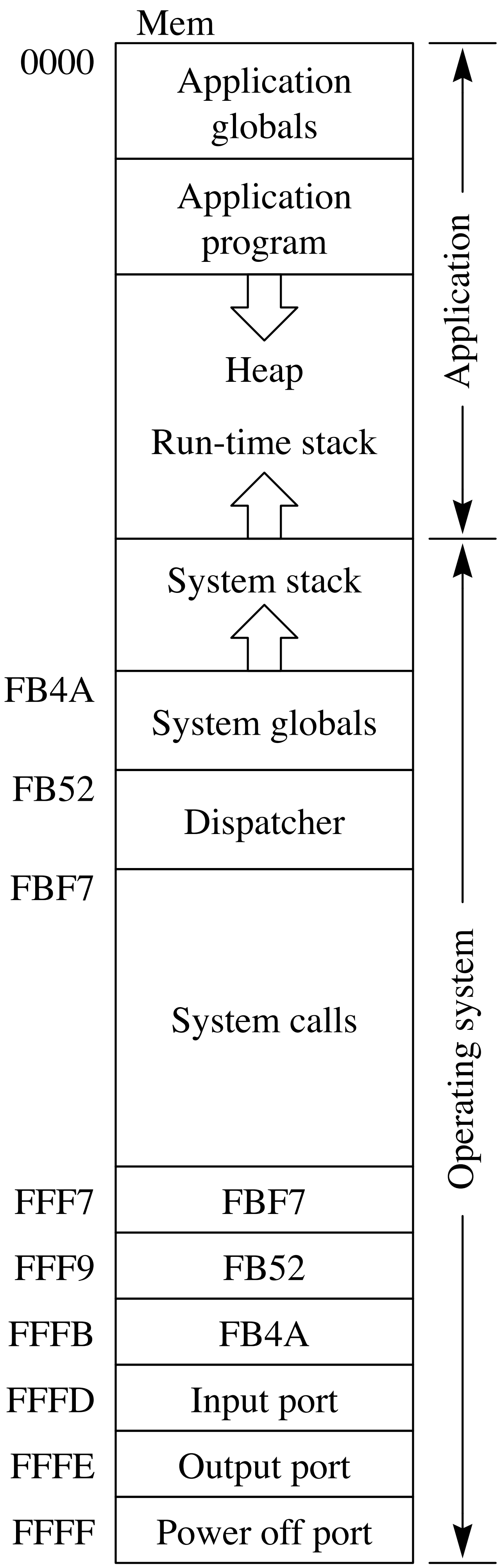
Compiling to the Assembly Level



(a) Translation directly to machine language.

(b) Translation to assembly language.

Figure 6.2



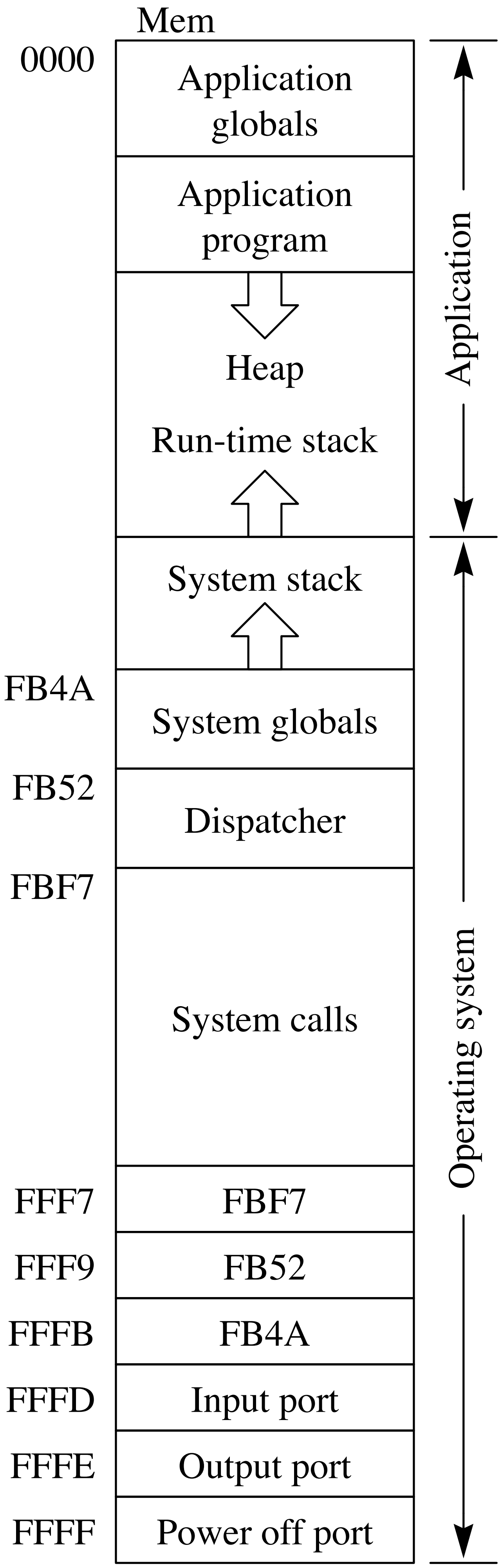
Mem

0000	Application globals
	Application program
	Heap
	Run-time stack
	System stack
FB4A	System globals
FB52	Dispatcher
FBF7	System calls
FFF7	FBF7
FFF9	FB52
FFFB	FB4A
FFFD	Input port
FFFE	Output port
FFFF	Power off port

Application

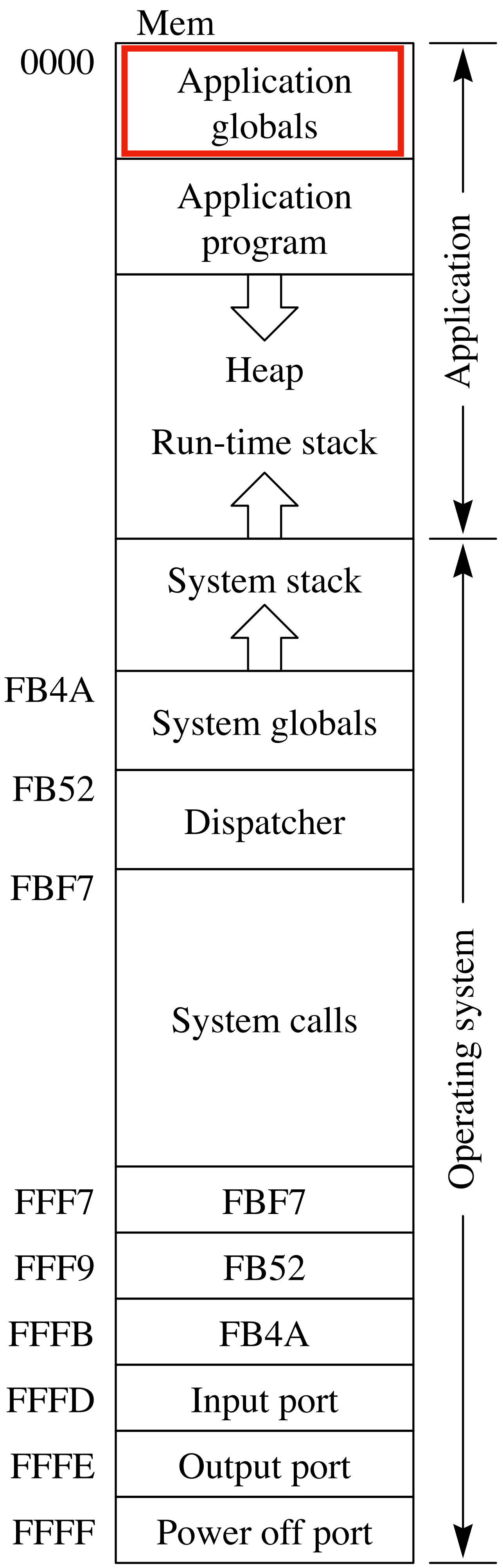
Operating system

Figure 6.2



- Global variables are stored at a fixed location in memory.
- Local variables and parameters are stored on the run-time stack.
- Dynamically allocated variables are stored on the heap.

Figure 6.2



- Global variables are stored at a fixed location in memory.
- Local variables and parameters are stored on the run-time stack.
- Dynamically allocated variables are stored on the heap.

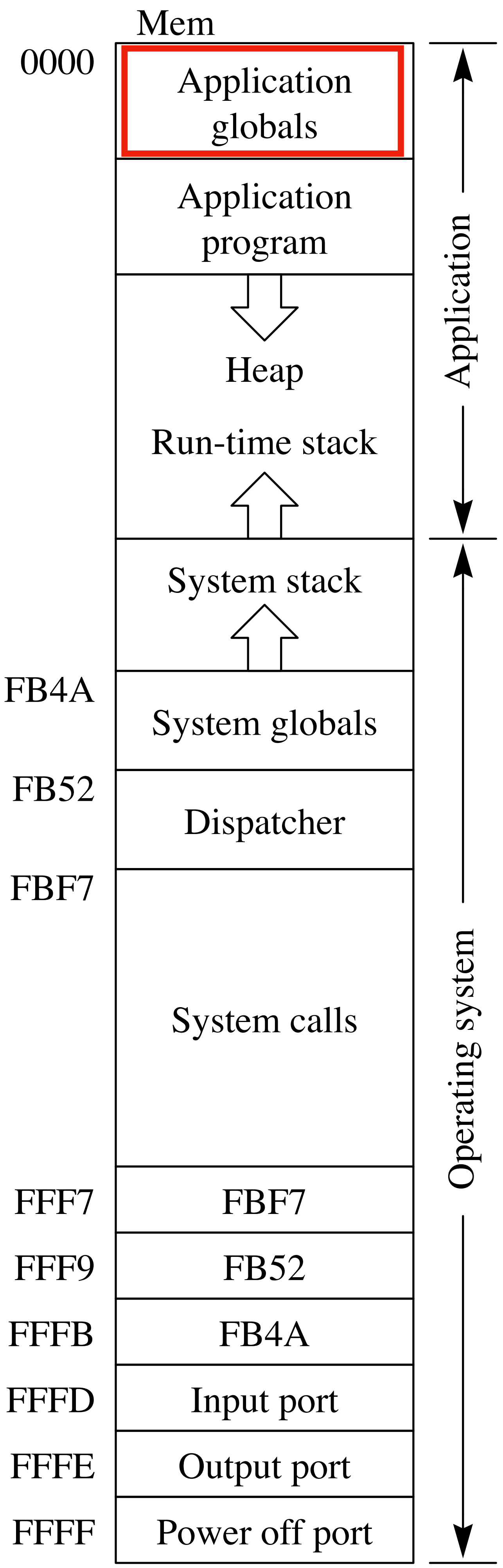


Figure 6.2

- Global variables are stored at a fixed location in memory.
- Local variables and parameters are stored on the run-time stack.
- Dynamically allocated variables are stored on the heap.

The unconditional branch instruction BR

High-Order Language

```
#include <stdio.h>
int age;

int main() {
    scanf(”%d”, &age);
    printf(”Age: %d years\n”, age);
    return 0;
}
```

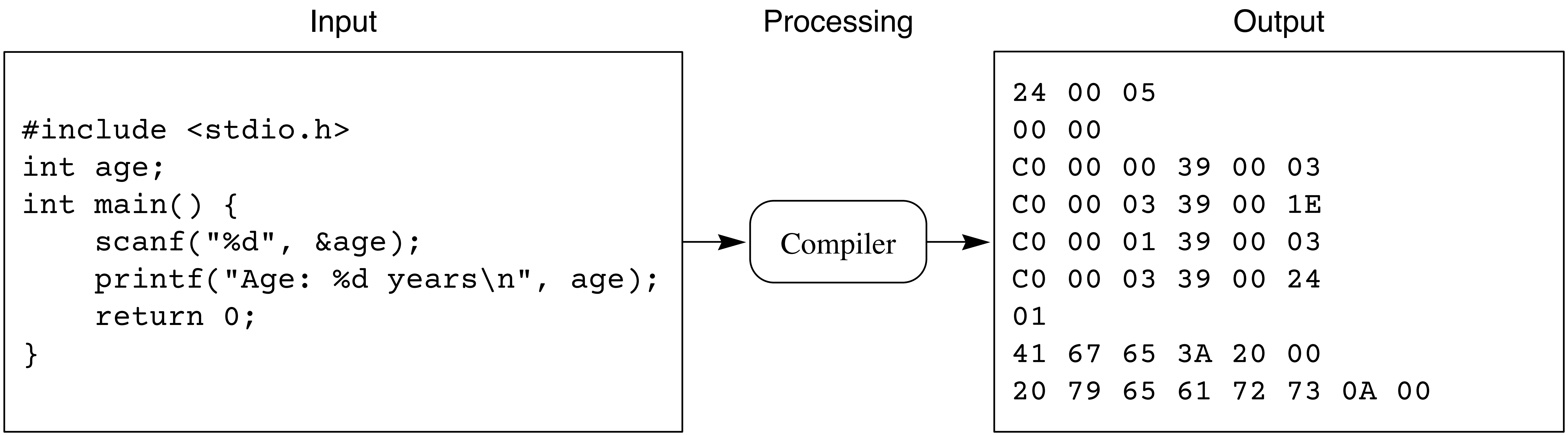
Figure 6.3

High-Order Language

```
#include <stdio.h>
int age;

int main() {
    scanf(″%d″, &age);
    printf(″Age: %d years\n″, age);
    return 0;
}
```

Assembly Language

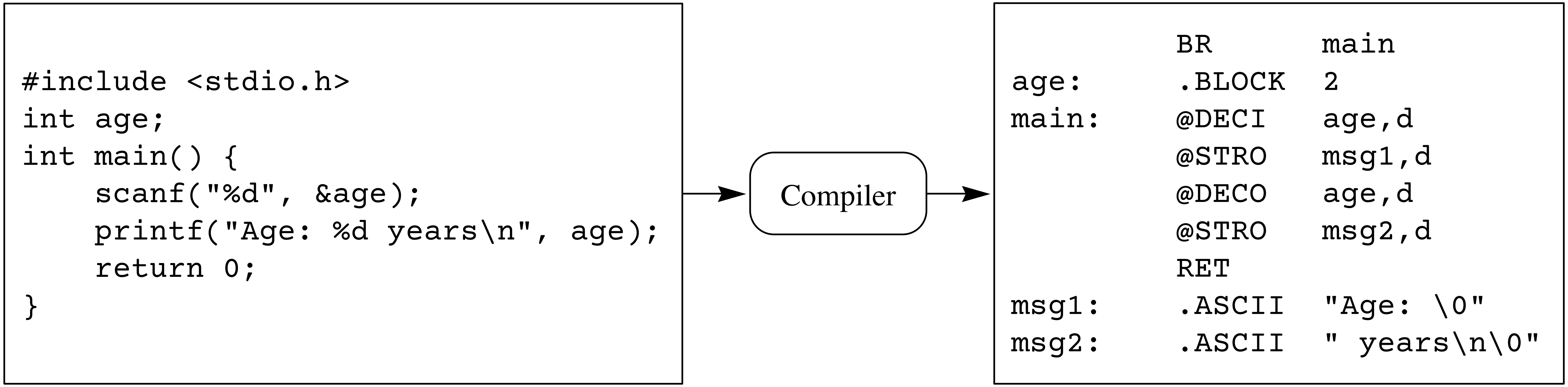
```
         BR      main
age:     .BLOCK  2            ;global variable #2d
;
main:    @DECI   age,d        ;scanf(″%d″, &age)
         @STRO   msg1,d       ;printf(″Age: %d years\n″, age)
         @DECO   age,d
         @STRO   msg2,d
         RET                  ;return 0
msg1:    .ASCII  ″Age: \0″
msg2:    .ASCII  ″ years\n\0″
```



```
#include <stdio.h>
int age;
int main() {
    scanf("%d", &age);
    printf("Age: %d years\n", age);
    return 0;
}
```

```
24 00 05
00 00
C0 00 00 39 00 03
C0 00 03 39 00 1E
C0 00 01 39 00 03
C0 00 03 39 00 24
01
41 67 65 3A 20 00
20 79 65 61 72 73 0A 00
```

(a) Translation directly to machine language.



```
#include <stdio.h>
int age;
int main() {
    scanf("%d", &age);
    printf("Age: %d years\n", age);
    return 0;
}
```

```
         BR       main
age:     .BLOCK   2
main:    @DECI    age,d
         @STRO    msg1,d
         @DECO    age,d
         @STRO    msg2,d
         RET
msg1:    .ASCII   "Age: \0"
msg2:    .ASCII   " years\n\0"
```

(b) Translation to assembly language.

```
         BR       main
age:     .BLOCK   2             ;global variable #2d
;
main:    @DECI    age,d         ;scanf(”%d”, &age)
         @STRO    msg1,d        ;printf(”Age: %d years\n”, age)
         @DECO    age,d
         @STRO    msg2,d
         RET                    ;return 0
msg1:    .ASCII   ”Age: \0”
msg2:    .ASCII   ” years\n\0”
```

```
         BR       main
age:     .BLOCK   2            ;global variable #2d
;
main:    @DECI    age,d        ;scanf(”%d”, &age)
         @STRO    msg1,d       ;printf(”Age: %d years\n”, age)
         @DECO    age,d
         @STRO    msg2,d
         RET                   ;return 0
msg1:    .ASCII   ”Age: \0”
msg2:    .ASCII   ” years\n\0”
```

Default addressing mode

```
BR       main,i
```

```
0000   240005             BR       main
0003   0000    age:       .BLOCK   2              ;global variable #2d
               ;main:     @DECI    age,d          ;scanf(”%d”, &age)
0005   C00000 main:       LDWA     DECI,i
0008   390003             SCALL    age,d
               ;          End @DECI
```

```
0000   240005               BR       main
0003   0000    age:         .BLOCK   2                ;global variable #2d
               ;main:       @DECI    age,d            ;scanf("%d", &age)
0005   C00000  main:        LDWA     DECI,i
0008   390003               SCALL    age,d
               ;            End @DECI
```

The BR instruction

$$\text{PC} \leftarrow \text{Oprnd}$$

```
0000   240005             BR      main
0003   0000    age:       .BLOCK  2              ;global variable #2d
               ;main:     @DECI   age,d          ;scanf("%d", &age)
0005   C00000 main:       LDWA    DECI,i
0008   390003             SCALL   age,d
               ;          End @DECI
```

The BR instruction

$$PC \leftarrow Oprnd$$

```
Load program into memory at Mem[0000]
$PC \leftarrow 0000$
do
    Fetch: $IR[0:7] \leftarrow Mem[PC]$
    Decode: Decode instruction specifier $IR[0:7]$
    Increment: $PC \leftarrow PC + 1$
    if $IR[0:7]$ is a dyadic instruction
        $IR[8:23] \leftarrow Mem[PC]$
        $PC \leftarrow PC + 2$
    Execute: Execute the instruction in the IR
while not shut down && instruction in IR is legal
```

```
0000   240005              BR      main
0003   0000    age:        .BLOCK  2            ;global variable #2d
               ;main:      @DECI   age,d        ;scanf("%d", &age)
0005   C00000  main:       LDWA    DECI,i
0008   390003              SCALL   age,d
               ;           End @DECI
```

The BR instruction

$$PC \leftarrow Oprnd$$

```
Load program into memory at Mem[0000]
PC ← 0000
do
    Fetch: IR[0:7] ← Mem[PC]
    Decode: Decode instruction specifier IR[0:7]
    Increment: PC ← PC + 1
    if IR[0:7] is a dyadic instruction
        IR[8:23] ← Mem[PC]
        PC ← PC + 2
    Execute: Execute the instruction in the IR
while not shut down && instruction in IR is legal
```

```
0000   240005              BR       main
0003   0000    age:        .BLOCK   2            ;global variable #2d
               ;main:      @DECI    age,d        ;scanf(″%d″, &age)
0005   C00000  main:       LDWA     DECI,i
0008   390003              SCALL    age,d
               ;           End @DECI
```

The BR instruction

PC ← Oprnd

```
Load program into memory at Mem[0000]
PC ← 0000
do
    Fetch: IR[0:7] ← Mem[PC]
    Decode: Decode instruction specifier IR[0:7]
    Increment: PC ← PC + 1
    if IR[0:7] is a dyadic instruction
        IR[8:23] ← Mem[PC]
        PC ← PC + 2
    Execute: Execute the instruction in the IR
while not shut down && instruction in IR is legal
```

```
0000   240005              BR       main
0003   0000    age:        .BLOCK   2              ;global variable #2d
               ;main:      @DECI    age,d          ;scanf(”%d”, &age)
0005   C00000  main:       LDWA     DECI,i
0008   390003              SCALL    age,d
               ;           End @DECI
```

The BR instruction

$$PC \leftarrow Oprnd$$

```
Load program into memory at Mem[0000]
PC ← 0000
do
    Fetch: IR[0:7] ← Mem[PC]
    Decode: Decode instruction specifier IR[0:7]
    Increment: PC ← PC + 1
    if IR[0:7] is a dyadic instruction
        IR[8:23] ← Mem[PC]
        PC ← PC + 2
    Execute: Execute the instruction in the IR
while not shut down && instruction in IR is legal
```

```
0000   240005             BR       main
0003   0000    age:       .BLOCK   2            ;global variable #2d
               ;main:     @DECI    age,d        ;scanf("%d", &age)
0005   C00000 main:       LDWA     DECI,i
0008   390003             SCALL    age,d
               ;          End @DECI
```

The BR instruction

$$PC \leftarrow Oprnd$$

Cycle	State
0	A: PC: 0000 IR:
1-fetch	
1-increment	
1-execute	
2-fetch	
2-increment	
2-execute	

```
0000   240005               BR       main
0003   0000    age:         .BLOCK   2             ;global variable #2d
               ;main:       @DECI    age,d         ;scanf("%d", &age)
0005   C00000  main:        LDWA     DECI,i
0008   390003               SCALL    age,d
               ;            End @DECI
```

The BR instruction

$$PC \leftarrow Oprnd$$

Cycle	State		
0	A:	PC: 0000	IR:
1-fetch	A:	PC: 0000	IR: 240005
1-increment			
1-execute			
2-fetch			
2-increment			
2-execute			

```
0000   240005             BR       main
0003   0000    age:       .BLOCK   2            ;global variable #2d
               ;main:     @DECI    age,d        ;scanf("%d", &age)
0005   C00000 main:       LDWA     DECI,i
0008   390003             SCALL    age,d
               ;          End @DECI
```

The BR instruction

$$PC \leftarrow Oprnd$$

Cycle	State		
0	A:	PC: 0000	IR:
1-fetch	A:	PC: 0000	IR: 240005
1-increment	A:	PC: 0003	IR: 240005
1-execute			
2-fetch			
2-increment			
2-execute			

```
0000   240005              BR       main
0003   0000    age:        .BLOCK   2             ;global variable #2d
               ;main:      @DECI    age,d         ;scanf("%d", &age)
0005   C00000  main:       LDWA     DECI,i
0008   390003              SCALL    age,d
               ;           End @DECI
```

The BR instruction

$$PC \leftarrow Oprnd$$

Cycle	State		
0	A:	PC: 0000	IR:
1-fetch	A:	PC: 0000	IR: 240005
1-increment	A:	PC: 0003	IR: 240005
1-execute	A:	PC: 0005	IR: 240005
2-fetch			
2-increment			
2-execute			

```
0000   240005            BR      main
0003   0000    age:      .BLOCK  2             ;global variable #2d
               ;main:    @DECI   age,d         ;scanf("%d", &age)
0005   C00000 main:      LDWA    DECI,i
0008   390003            SCALL   age,d
               ;         End @DECI
```

The BR instruction

$$PC \leftarrow Oprnd$$

Cycle	State		
0	A:	PC: 0000	IR:
1-fetch	A:	PC: 0000	IR: 240005
1-increment	A:	PC: 0003	IR: 240005
1-execute	A:	PC: 0005	IR: 240005
2-fetch	A:	PC: 0005	IR: C00000
2-increment			
2-execute			

```
0000   240005              BR       main
0003   0000    age:        .BLOCK   2              ;global variable #2d
               ;main:      @DECI    age,d          ;scanf("%d", &age)
0005   C00000  main:       LDWA     DECI,i
0008   390003              SCALL    age,d
               ;           End @DECI
```

The BR instruction

$$PC \leftarrow Oprnd$$

Cycle	State		
0	A:	PC: 0000	IR:
1-fetch	A:	PC: 0000	IR: 240005
1-increment	A:	PC: 0003	IR: 240005
1-execute	A:	PC: 0005	IR: 240005
2-fetch	A:	PC: 0005	IR: C00000
2-increment	A:	PC: 0008	IR: C00000
2-execute			

```
0000   240005              BR       main
0003   0000    age:        .BLOCK   2             ;global variable #2d
               ;main:      @DECI    age,d         ;scanf(”%d”, &age)
0005   C00000  main:       LDWA     DECI,i
0008   390003              SCALL    age,d
               ;           End @DECI
```

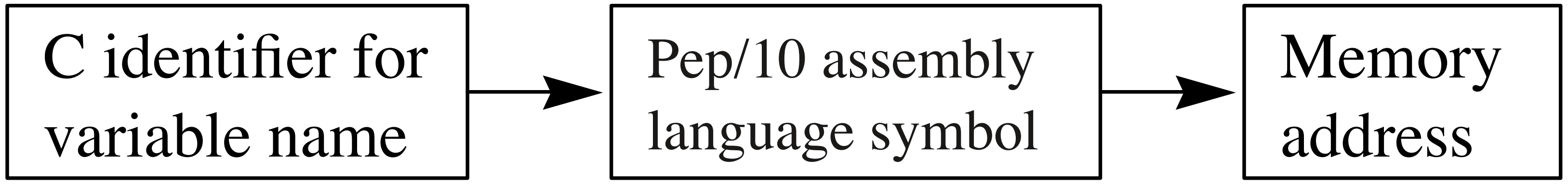
The BR instruction

$$\text{PC} \leftarrow \text{Oprnd}$$

Cycle	State		
0	A:	PC: 0000	IR:
1-fetch	A:	PC: 0000	IR: 240005
1-increment	A:	PC: 0003	IR: 240005
1-execute	A:	PC: 0005	IR: 240005
2-fetch	A:	PC: 0005	IR: C00000
2-increment	A:	PC: 0008	IR: C00000
2-execute	A: 0000	PC: 0008	IR: C00000

High-Order Language

```
#include <stdio.h>
int age;

int main() {
    scanf(″%d″, &age);
    printf(″Age: %d years\n″, age);
    return 0;
}
```



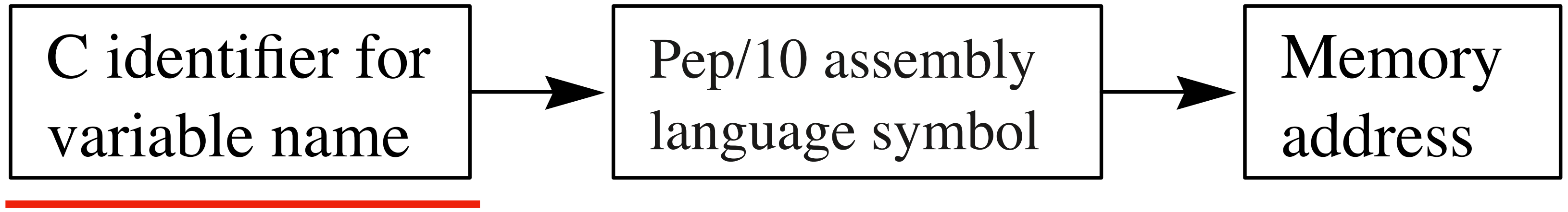
C identifier for variable name → Pep/10 assembly language symbol → Memory address

Assembly Language

```
         BR      main
age:     .BLOCK  2           ;global variable #2d
;
main:    @DECI   age,d       ;scanf(″%d″, &age)
         @STRO   msg1,d      ;printf(″Age: %d years\n″, age)
         @DECO   age,d
         @STRO   msg2,d
         RET                 ;return 0
msg1:    .ASCII  ″Age: \0″
msg2:    .ASCII  ″ years\n\0″
```

High-Order Language

```
#include <stdio.h>
int age;

int main() {
    scanf("%d", &age);
    printf("Age: %d years\n", age);
    return 0;
}
```



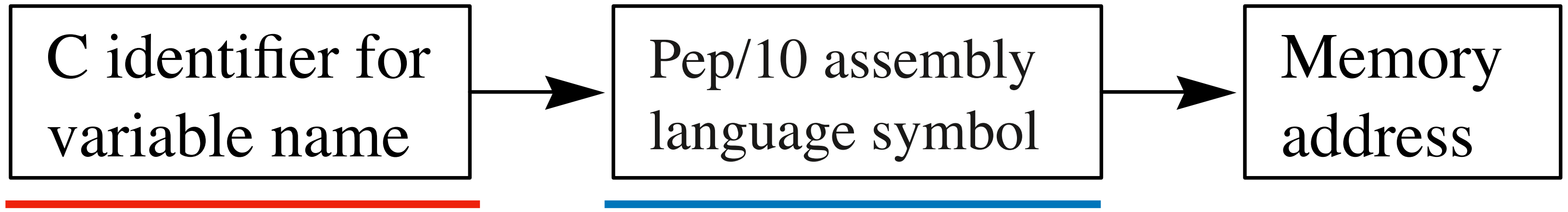
C identifier for variable name → Pep/10 assembly language symbol → Memory address

Assembly Language

```
         BR       main
age:     .BLOCK   2            ;global variable #2d
;
main:    @DECI    age,d        ;scanf("%d", &age)
         @STRO    msg1,d       ;printf("Age: %d years\n", age)
         @DECO    age,d
         @STRO    msg2,d
         RET                   ;return 0
msg1:    .ASCII   "Age: \0"
msg2:    .ASCII   " years\n\0"
```

High-Order Language

```
#include <stdio.h>
int age;

int main() {
    scanf("%d", &age);
    printf("Age: %d years\n", age);
    return 0;
}
```


C identifier for variable name
Pep/10 assembly language symbol
Memory address

Assembly Language

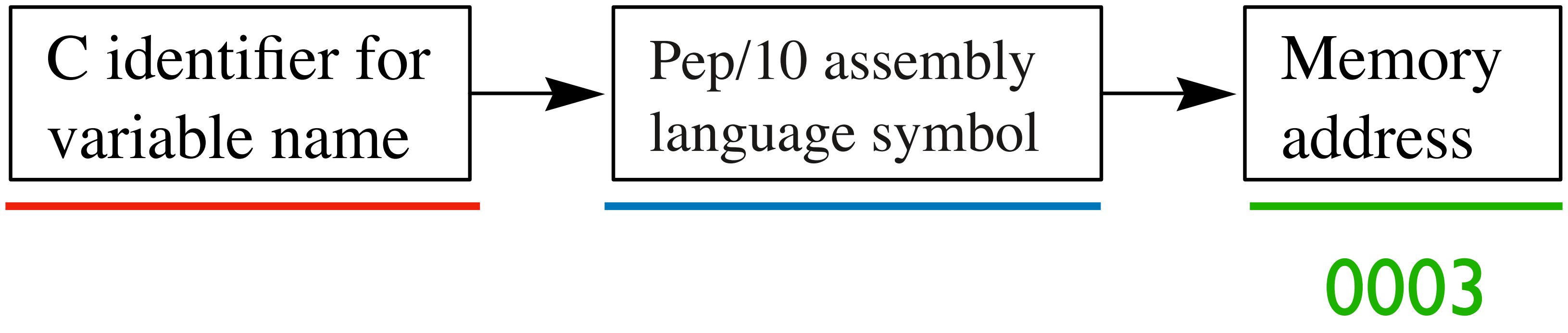
```
         BR      main
age:     .BLOCK  2            ;global variable #2d
;
main:    @DECI   age,d        ;scanf("%d", &age)
         @STRO   msg1,d       ;printf("Age: %d years\n", age)
         @DECO   age,d
         @STRO   msg2,d
         RET                  ;return 0
msg1:    .ASCII  "Age: \0"
msg2:    .ASCII  " years\n\0"
```

Computer Systems Sixth edition

Figure 6.6

High-Order Language

```
#include <stdio.h>
int age;

int main() {
    scanf("%d", &age);
    printf("Age: %d years\n", age);
    return 0;
}
```



C identifier for variable name
Pep/10 assembly language symbol
Memory address
0003

Assembly Language

```
         BR      main
age:     .BLOCK  2            ;global variable #2d
;
main:    @DECI   age,d        ;scanf("%d", &age)
         @STRO   msg1,d       ;printf("Age: %d years\n", age)
         @DECO   age,d
         @STRO   msg2,d
         RET                  ;return 0
msg1:    .ASCII  "Age: \0"
msg2:    .ASCII  " years\n\0"
```

High-Order Language

```
#include <stdio.h>
int age;

int main() {
    scanf("%d", &age);
    printf("Age: %d years\n", age);
    return 0;
}
```

Symbol	Scope	Type
age	global	int

(a) The compiler symbol table.

Symbol	Value
age	0003

(b) The assembler symbol table

Assembly Language

```
         BR       main
age:     .BLOCK   2            ;global variable #2d
;
main:    @DECI    age,d        ;scanf("%d", &age)
         @STRO    msg1,d       ;printf("Age: %d years\n", age)
         @DECO    age,d
         @STRO    msg2,d
         RET                   ;return 0
msg1:    .ASCII   "Age: \0"
msg2:    .ASCII   " years\n\0"
```

Translating global variables

- Allocate global variables at a fixed location in memory with `.BLOCK`.
- Access global variables with direct addressing, `d`.

Translating `scanf()`

- Translate character input with `@CHARI`.
- Translate integer input with `@DECI`.

Translating `printf()`

- Translate character output with `@CHARO`.
- Translate integer output with `@DECO`.
- Translate string output of any substrings from the format string with `@STRO` or `@CHARO`.

The Pep/10 Symbol tracer

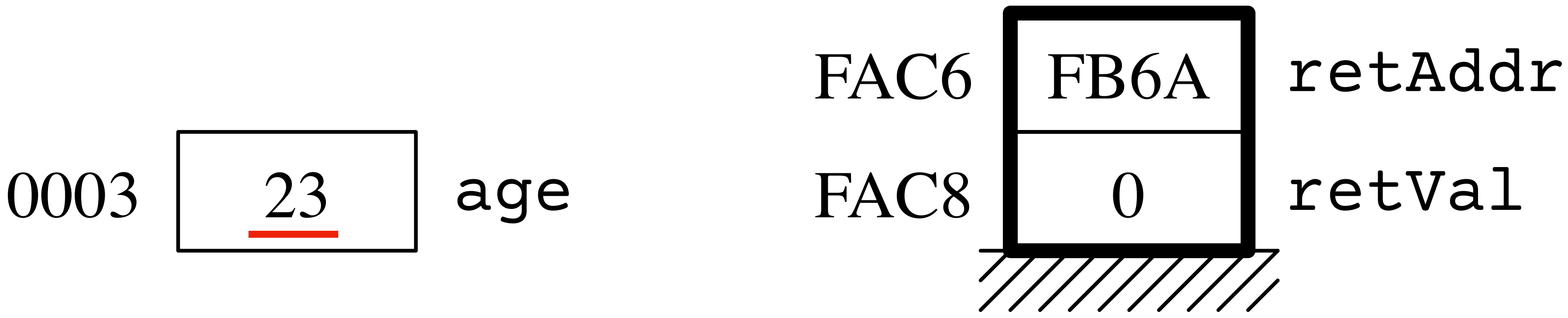
- Format trace tags
  - Required for global and local variables
- Symbol trace tags
  - Not required for global variables

Format trace tags

`#1c`	One-byte character
`#1d`	One-byte signed decimal
`#2d`	Two-byte signed decimal
`#1u`	One-byte unsigned decimal
`#2u`	Two-byte unsigned decimal
`#1h`	One-byte hexadecimal
`#2h`	Two-byte hexadecimal

Assembly Language

```
         BR      main
age:     .BLOCK  2            ;global variable #2d
;
main:    @DECI   age,d        ;scanf("%d", &age)
         @STRO   msg1,d       ;printf("Age: %d years\n", age)
         @DECO   age,d
         @STRO   msg2,d
         RET                  ;return 0
msg1:    .ASCII  "Age: \0"
msg2:    .ASCII  " years\n\0"
```



High-Order Language

```
#include <stdio.h>
const int bonus = 10;
int exam1;
int exam2;
int score;

int main() {
    scanf(”%d %d”, &exam1, &exam2);
    score = (exam1 + exam2) / 2 + bonus;
    printf(”score = %d\n”, score);
    return 0;
}
```

Assembly Language

```
         BR      main
bonus:   .EQUATE 10            ;constant
exam1:   .BLOCK  2             ;global variable #2d
exam2:   .BLOCK  2             ;global variable #2d
score:   .BLOCK  2             ;global variable #2d
;
main:    @DECI   exam1,d       ;scanf(”%d %d”, &exam1, &exam2)
         @DECI   exam2,d
         LDWA    exam1,d       ;score = (exam1 + exam2) / 2 + bonus
         ADDA    exam2,d
         ASRA
         ADDA    bonus,i
         STWA    score,d
         @STRO   msg,d         ;printf(”score = %d\n”, score)
         @DECO   score,d
         @CHARO  ’\n’,i
         RET
msg:     .ASCII  ”score = \0”
```

Constants

- Equate the constant to its value with `.EQUATE`.
- `.EQUATE` does not generate object code.
- The value of the constant symbol is not an address.
- Assess the constant with immediate addressing, `i`.

Constants

```
0000  240009                BR      main
             bonus:   .EQUATE 10          ;constant
0003  0000   exam1:   .BLOCK  2           ;global variable #2d
0005  0000   exam2:   .BLOCK  2           ;global variable #2d
0007  0000   score:   .BLOCK  2           ;global variable #2d


001C  50000A                ADDA    bonus,i
```

Constants

```
0000  240009            BR       main
             bonus:     .EQUATE 10              ;constant
0003  0000   exam1:     .BLOCK   2              ;global variable #2d
0005  0000   exam2:     .BLOCK   2              ;global variable #2d
0007  0000   score:     .BLOCK   2              ;global variable #2d


001C  50000A            ADDA     bonus,i
```

Symbol	Value
bonus	
exam1	
exam2	
score	

Constants

```
0000  240009            BR      main
               bonus:   .EQUATE 10          ;constant
0003  0000     exam1:   .BLOCK  2           ;global variable #2d
0005  0000     exam2:   .BLOCK  2           ;global variable #2d
0007  0000     score:   .BLOCK  2           ;global variable #2d


001C  50000A            ADDA    bonus,i
```

Symbol	Value
bonus	
exam1	
exam2	
score	0007

Constants

```
0000   240009             BR       main
                bonus:    .EQUATE 10                ;constant
0003   0000     exam1:    .BLOCK   2                ;global variable #2d
0005   0000     exam2:    .BLOCK   2                ;global variable #2d
0007   0000     score:    .BLOCK   2                ;global variable #2d


001C   50000A             ADDA     bonus,i
```

Symbol	Value
bonus	
exam1	
exam2	0005
score	0007

Constants

```
0000   240009                 BR       main
                bonus:       .EQUATE  10               ;constant
0003   0000     exam1:       .BLOCK   2                ;global variable #2d
0005   0000     exam2:       .BLOCK   2                ;global variable #2d
0007   0000     score:       .BLOCK   2                ;global variable #2d


001C   50000A                 ADDA     bonus,i
```

Symbol	Value
bonus	
exam1	0003
exam2	0005
score	0007

Constants

```
0000   240009              BR       main
              bonus:       .EQUATE  10               ;constant
0003   0000   exam1:       .BLOCK   2                ;global variable #2d
0005   0000   exam2:       .BLOCK   2                ;global variable #2d
0007   0000   score:       .BLOCK   2                ;global variable #2d


001C   50000A              ADDA     bonus,i
```

Symbol	Value
bonus	000A
exam1	0003
exam2	0005
score	0007

Assignment statements

variable = expression ;

- Load the accumulator from the expression with `LDWA` for type `int` or `LDBA` for type `char`.
- Compute the value of the expression if necessary.
- Store the value to the variable with `STWA` or `STBA`.

Immediate addressing

- Oprnd = OprndSpec
- Asmb5 letter: `i`
- The operand specifier *is* the operand.

Direct addressing

- Oprnd = Mem[OprndSpec]
- Asmb5 letter: `d`
- The operand specifier is the *address* in memory of the operand.

Stack-relative addressing

- Oprnd = Mem[SP + OprndSpec]
- Asmb5 letter: s
- The stack pointer *plus* the operand specifier is the *address* in memory of the operand.

The add SP instruction

- Instruction specifier: 0100 0aaa
- Mnemonic: `ADDSP`
- Adds the operand to the stack pointer, SP

$$\mathrm{SP} \leftarrow \mathrm{SP} + \mathrm{Oprnd}$$

The subtract SP

- Instruction specifier: 0100 1aaa
- Mnemonic: `SUBSP`
- Subtracts the operand from the stack pointer, SP

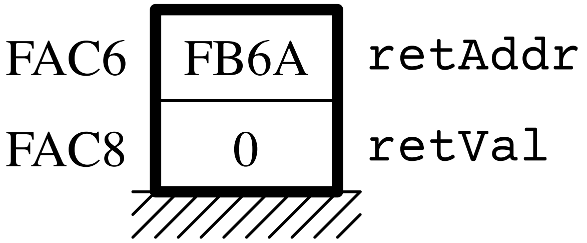
$$\mathrm{SP} \leftarrow \mathrm{SP} - \mathrm{Oprnd}$$

```
LDBA     ’B’,i        ;move ’B’ to stack
STBA     -1,s
LDBA     ’M’,i        ;move ’M’ to stack
STBA     -2,s
LDBA     ’W’,i        ;move ’W’ to stack
STBA     -3,s
LDWA     335,i        ;move 335 to stack
STWA     -5,s
LDBA     ’i’,i        ;move ’i’ to stack
STBA     -6,s
SUBSP    6,i          ;push 6 bytes onto stack
@CHARO   5,s          ;output B
@CHARO   4,s          ;output M
@CHARO   3,s          ;output W
@DECO    1,s          ;output 335
@CHARO   0,s          ;output i
ADDSP    6,i          ;pop 6 bytes off stack
RET
```

Output

```
BMW335i
```

Figure 6.11



FAC6	FB6A	`retAddr`
FAC8	0	`retVal`

(a)

SP: FAC6
Initial state

Computer Systems Sixth edition

Figure 6.11

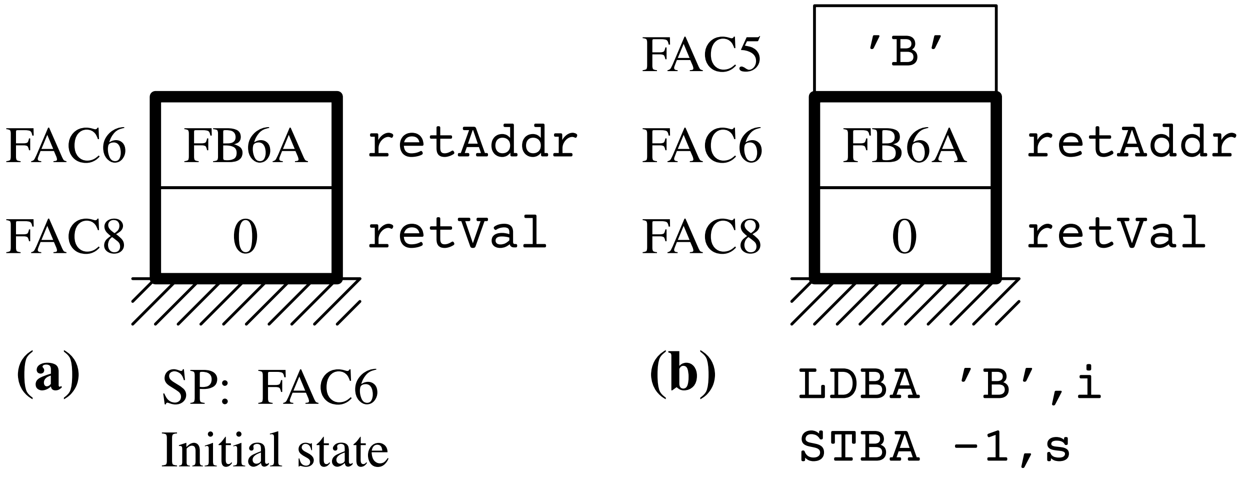
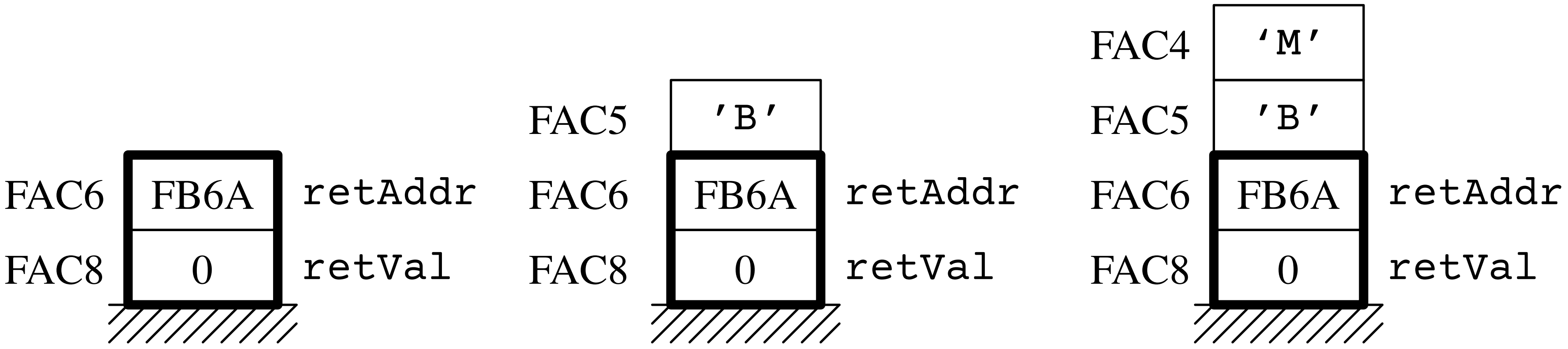


Figure 6.11



(a) SP: FAC6
Initial state

(b)

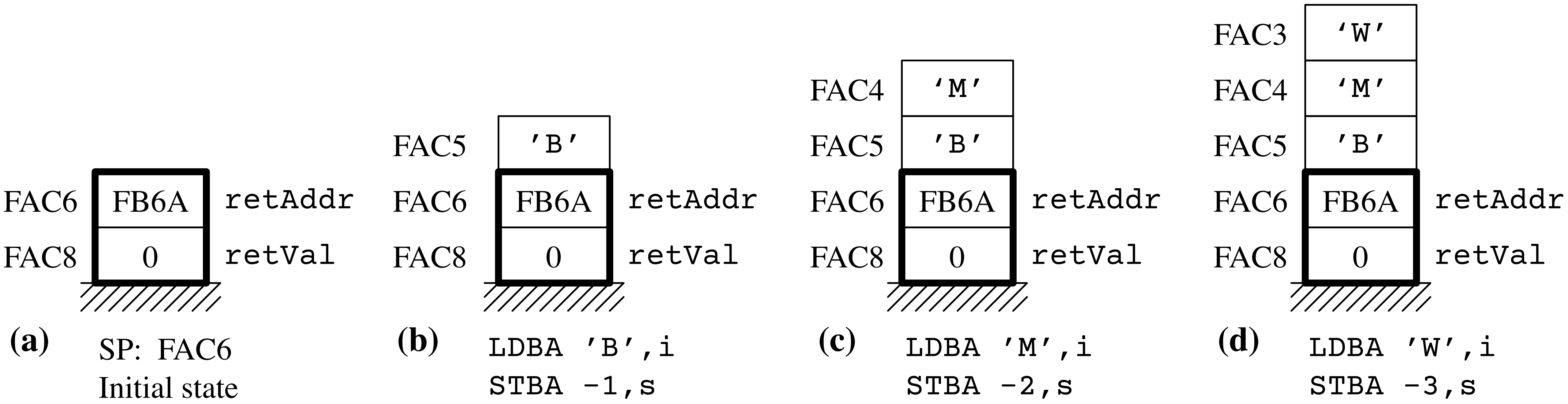
```
LDBA ’B’,i
STBA -1,s
```

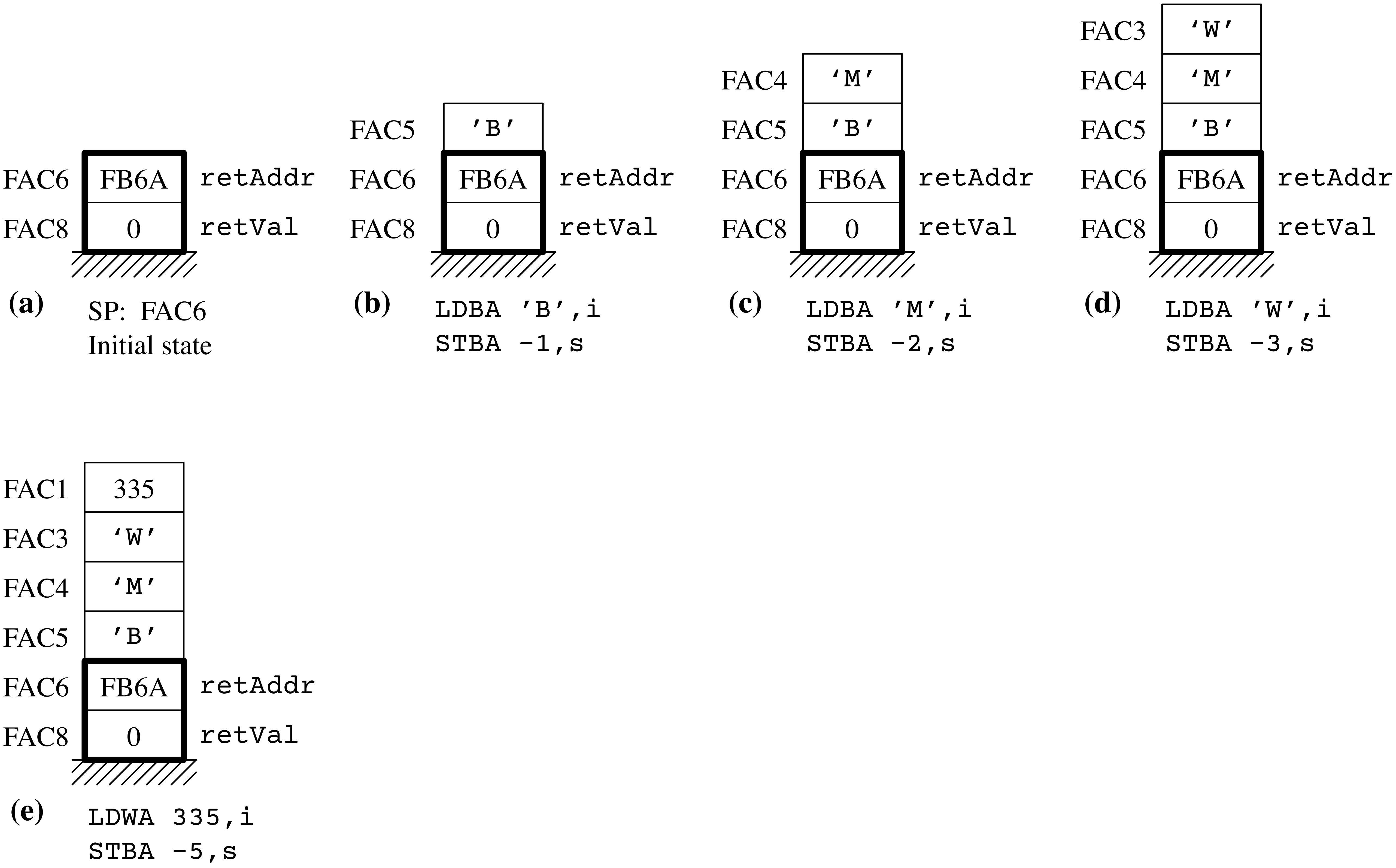
(c)

```
LDBA ’M’,i
STBA -2,s
```

Computer Systems Sixth edition

Figure 6.11





(a) SP: FAC6
Initial state

(b)
```
LDBA 'B',i
STBA -1,s
```

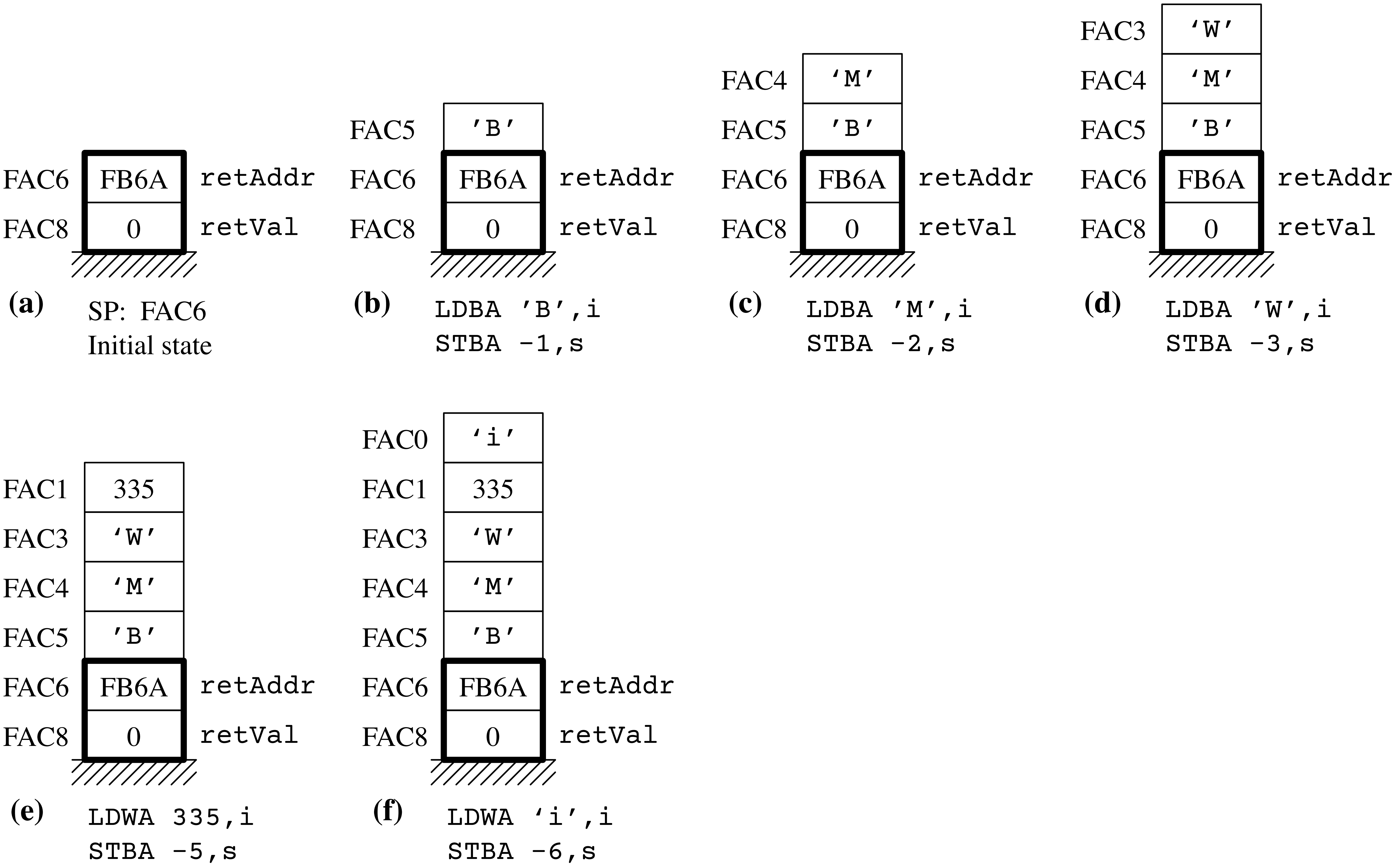
(c)
```
LDBA 'M',i
STBA -2,s
```

(d)
```
LDBA 'W',i
STBA -3,s
```

(e)
```
LDWA 335,i
STBA -5,s
```

Computer Systems Sixth edition

Figure 6.11



FAC6	FB6A	retAddr
FAC8	0	retVal

(a) SP: FAC6
Initial state

FAC5	’B’	
FAC6	FB6A	retAddr
FAC8	0	retVal

(b)
```
LDBA ’B’,i
STBA -1,s
```

FAC4	‘M’	
FAC5	’B’	
FAC6	FB6A	retAddr
FAC8	0	retVal

(c)
```
LDBA ’M’,i
STBA -2,s
```

FAC3	‘W’	
FAC4	‘M’	
FAC5	’B’	
FAC6	FB6A	retAddr
FAC8	0	retVal

(d)
```
LDBA ’W’,i
STBA -3,s
```

FAC1	335	
FAC3	‘W’	
FAC4	‘M’	
FAC5	’B’	
FAC6	FB6A	retAddr
FAC8	0	retVal

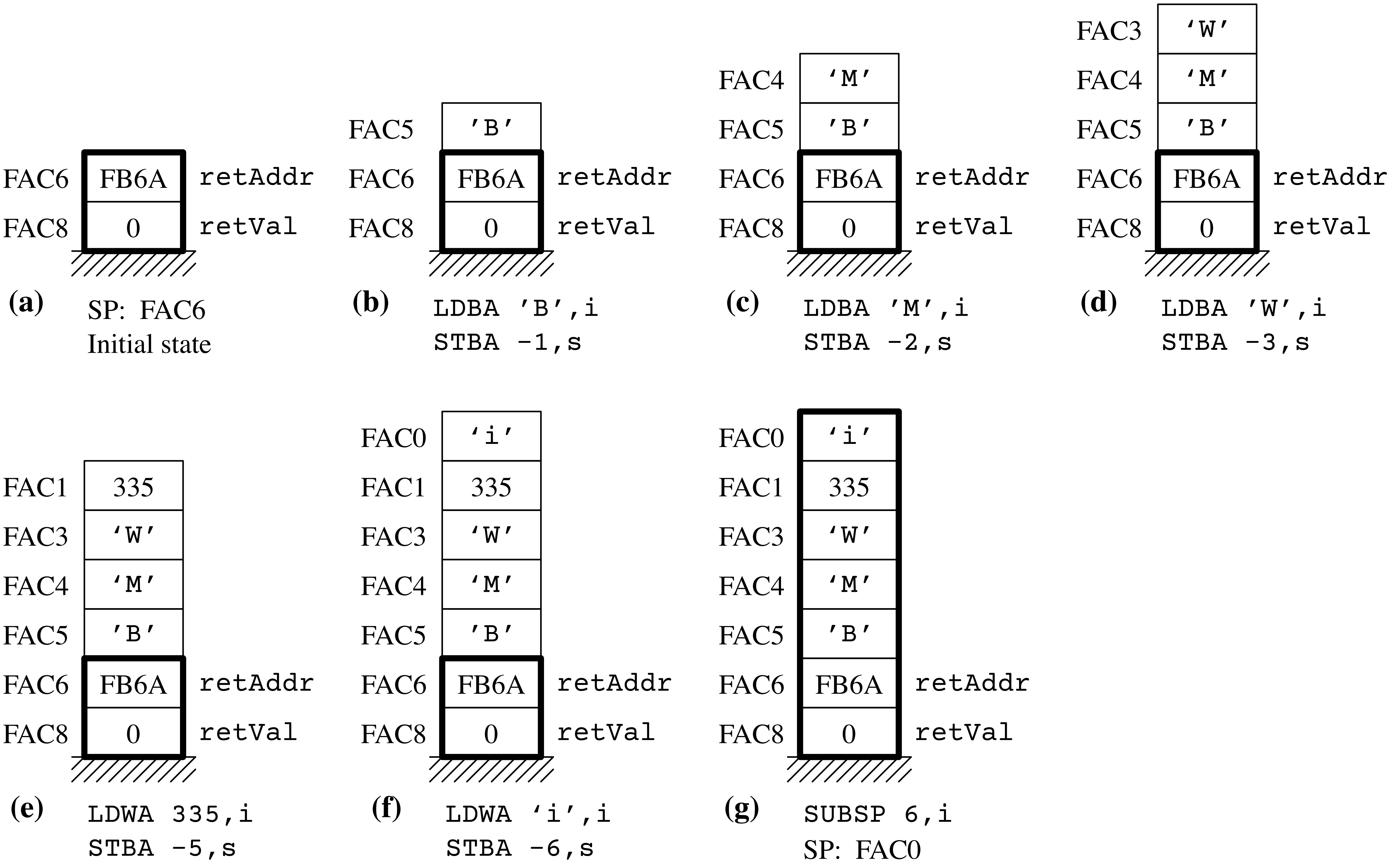
(e)
```
LDWA 335,i
STBA -5,s
```

FAC0	‘i’	
FAC1	335	
FAC3	‘W’	
FAC4	‘M’	
FAC5	’B’	
FAC6	FB6A	retAddr
FAC8	0	retVal

(f)
```
LDWA ‘i’,i
STBA -6,s
```

Computer Systems Sixth edition

Figure 6.11



FAC6 FB6A retAddr
FAC8 0 retVal

(a) SP: FAC6
Initial state

FAC5 'B'
FAC6 FB6A retAddr
FAC8 0 retVal

(b) `LDBA 'B',i`
`STBA -1,s`

FAC4 'M'
FAC5 'B'
FAC6 FB6A retAddr
FAC8 0 retVal

(c) `LDBA 'M',i`
`STBA -2,s`

FAC3 'W'
FAC4 'M'
FAC5 'B'
FAC6 FB6A retAddr
FAC8 0 retVal

(d) `LDBA 'W',i`
`STBA -3,s`

FAC1 335
FAC3 'W'
FAC4 'M'
FAC5 'B'
FAC6 FB6A retAddr
FAC8 0 retVal

(e) `LDWA 335,i`
`STBA -5,s`

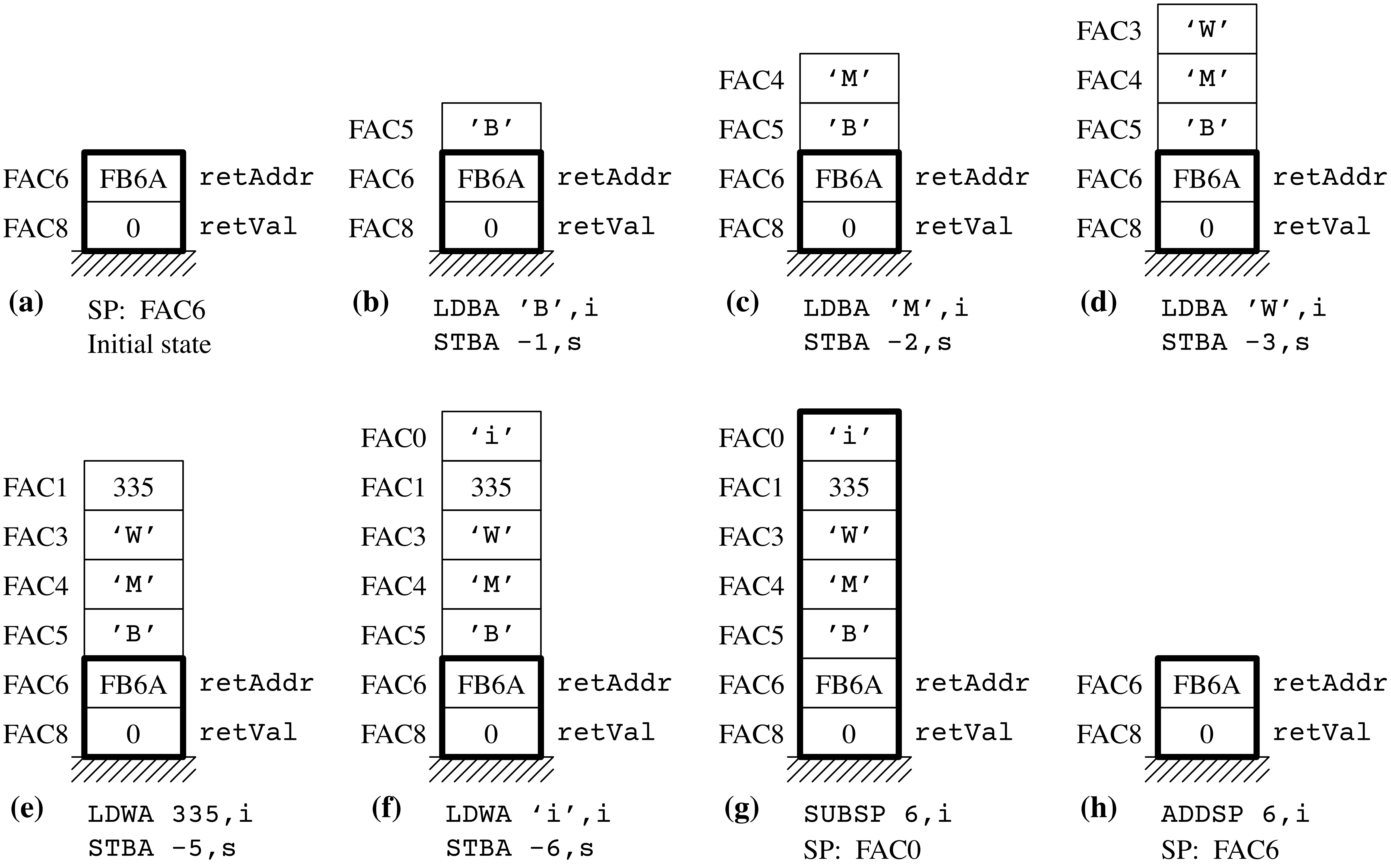
FAC0 'i'
FAC1 335
FAC3 'W'
FAC4 'M'
FAC5 'B'
FAC6 FB6A retAddr
FAC8 0 retVal

(f) `LDWA 'i',i`
`STBA -6,s`

FAC0 'i'
FAC1 335
FAC3 'W'
FAC4 'M'
FAC5 'B'
FAC6 FB6A retAddr
FAC8 0 retVal

(g) `SUBSP 6,i`
SP: FAC0

Figure 6.11



(a) SP: FAC6
Initial state

(b) LDBA ’B’,i
STBA -1,s

(c) LDBA ’M’,i
STBA -2,s

(d) LDBA ’W’,i
STBA -3,s

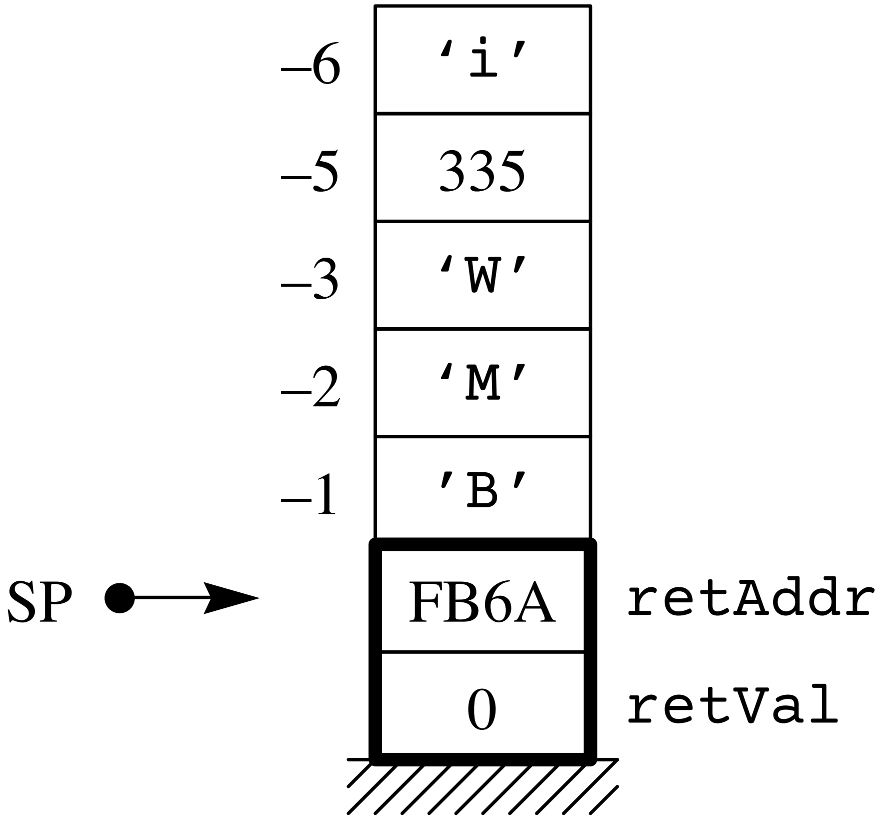
(e) LDWA 335,i
STBA -5,s

(f) LDWA ‘i’,i
STBA -6,s

(g) SUBSP 6,i
SP: FAC0

(h) ADDSP 6,i
SP: FAC6

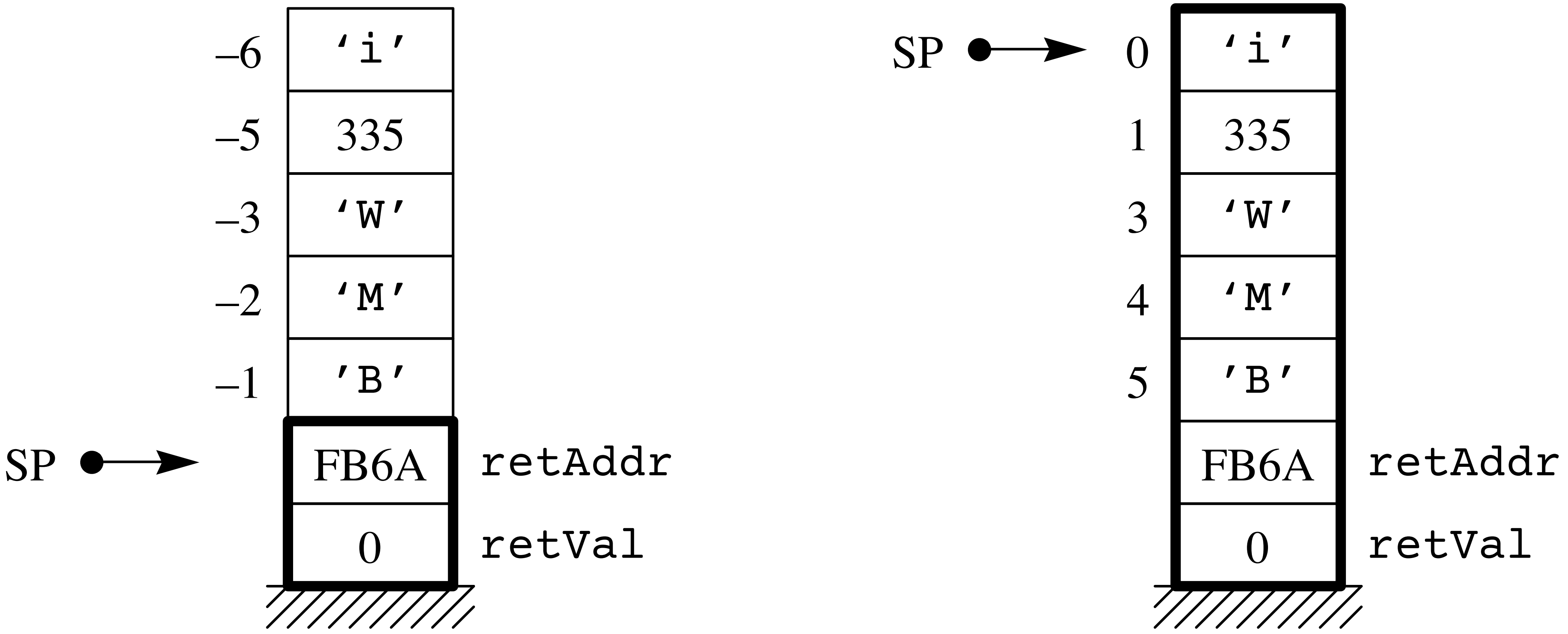
Stack-relative addresses



(f)

```
LDWA ‘i’,i
STBA -6,s
```

Stack-relative addresses



(f) `LDWA ‘i’,i`
`STBA -6,s`

(g) `SUBSP 6,i`
SP: FAC0

Local variables

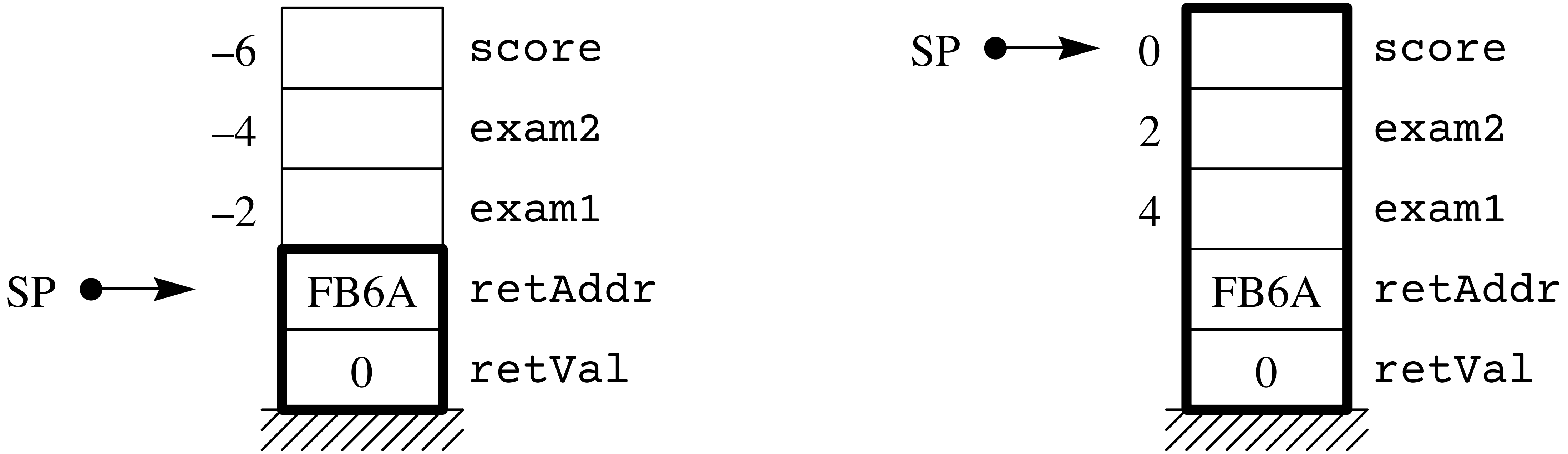
- Push locals with `SUBSP`
- Access locals with stack-relative addressing (`s`)
- Pop locals with `ADDSP`

High-Order Language

```
#include <stdio.h>

int main() {
    const int bonus = 10;
    int exam1;
    int exam2;
    int score;
    scanf(”%d %d”, &exam1, &exam2);
    score = (exam1 + exam2) / 2 + bonus;
    printf(”score = %d\n”, score);
    return 0;
}
```

Assembly Language

```
         BR       main
bonus:   .EQUATE  10          ;constant
exam1:   .EQUATE  4           ;local variable #2d
exam2:   .EQUATE  2           ;local variable #2d
score:   .EQUATE  0           ;local variable #2d
;
main:    SUBSP    6,i         ;push #exam1 #exam2 #score
         @DECI    exam1,s     ;scanf(”%d %d”, &exam1, &exam2)
         @DECI    exam2,s
         LDWA     exam1,s     ;score = (exam1 + exam2) / 2 + bonus
         ADDA     exam2,s
         ASRA
         ADDA     bonus,i
         STWA     score,s
         @STRO    msg,d       ;printf(”score = %d\n”, score)
         @DECO    score,s
         @CHARO   ’\n’,i
         ADDSP    6,i         ;pop #score #exam2 #exam1
         RET
msg:     .ASCII   ”score = \0”
```



(a) Before pushing locals.

(b) After pushing locals.

Return from main( )

```
#include <stdio.h>

int main(void) {
    return 666;
}
```

Assembly Language Listing

```
0000  240003             BR      main
             retVal:     .EQUATE 2
0003  C0029A main:       LDWA    666,i
0006  E30002             STWA    retVal,s
0009  01                 RET
```

Output

```
Main failed with return value 666
```

Return from `main( )`

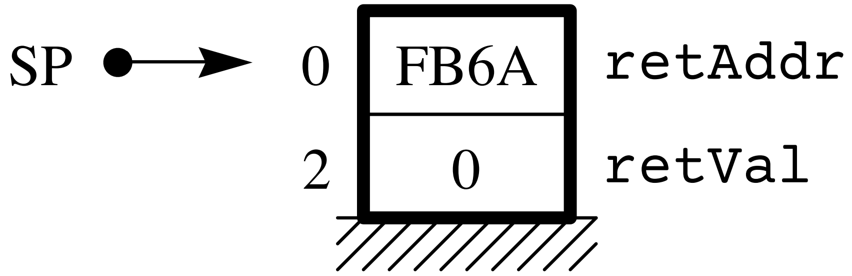
```
#include <stdio.h>

int main(void) {
    return 666;
}
```

Assembly Language Listing

```
0000  240003          BR      main
             retVal:  .EQUATE 2
0003  C0029A main:    LDWA    666,i
0006  E30002          STWA    retVal,s
0009  01              RET
```

Output

```
Main failed with return value 666
```



(a) Before `STWA retVal,s`.

Return from main( )

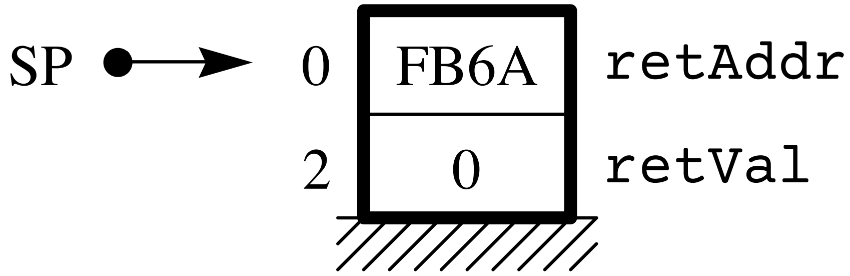
```
#include <stdio.h>

int main(void) {
    return 666;
}
```

Assembly Language Listing

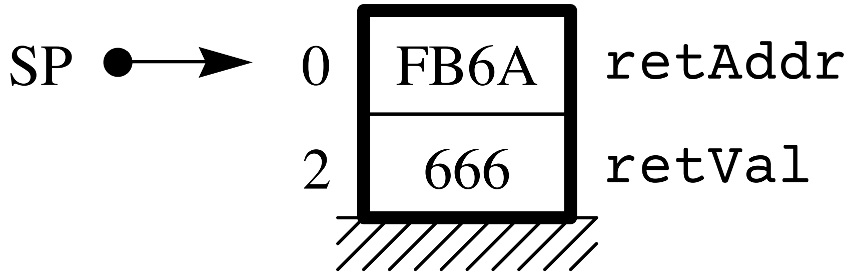
```
0000  240003         BR      main
             retVal: .EQUATE 2
0003  C0029A main:   LDWA    666,i
0006  E30002         STWA    retVal,s
0009  01             RET
```

Output

```
Main failed with return value 666
```



(a) Before `STWA retVal,s`.



(b) After `STWA retVal,s`.

Branching instructions

• `BRLE`	Branch on less than or equal to
• `BRLT`	Branch on less than
• `BREQ`	Branch on equal to
• `BRNE`	Branch on not equal to
• `BRGE`	Branch on greater than or equal to
• `BRGT`	Branch on greater than
• `BRV`	Branch on V
• `BRC`	Branch on C

Branching instructions

- `BRLE` $N = 1 \vee Z = 1 \Rightarrow \mathrm{PC} \leftarrow \mathrm{Oprnd}$
- `BRLT` $N = 1 \Rightarrow \mathrm{PC} \leftarrow \mathrm{Oprnd}$
- `BREQ` $Z = 1 \Rightarrow \mathrm{PC} \leftarrow \mathrm{Oprnd}$
- `BRNE` $Z = 0 \Rightarrow \mathrm{PC} \leftarrow \mathrm{Oprnd}$
- `BRGE` $N = 0 \Rightarrow \mathrm{PC} \leftarrow \mathrm{Oprnd}$
- `BRGT` $N = 0 \wedge Z = 0 \Rightarrow \mathrm{PC} \leftarrow \mathrm{Oprnd}$
- `BRV` $V = 1 \Rightarrow \mathrm{PC} \leftarrow \mathrm{Oprnd}$
- `BRC` $C = 1 \Rightarrow \mathrm{PC} \leftarrow \mathrm{Oprnd}$

```
#include <stdio.h>

int main() {
    int number;
    scanf(”%d”, &number);
    if (number < 0) {
        number = -number;
    }
    printf(”%d”, number);
    return 0;
}
```

Figure 6.17

```
         BR      main
number:  .EQUATE 0           ;local variable #2d
;
main:    SUBSP   2,i         ;push #number
         @DECI   number,s    ;scanf(”%d”, &number)
if:      LDWA    number,s    ;if (number < 0)
         BRGE    endIf
         LDWA    number,s    ;number = -number
         NEGA
         STWA    number,s
endIf:   @DECO   number,s    ;printf(”%d”, number)
         ADDSP   2,i         ;pop #number
         RET
         BR      main
number:  .EQUATE 0           ;local variable #2d
;
main:    SUBSP   2,i         ;push #number
         @DECI   number,s    ;scanf("%d", &number)
if:      LDWA    number,s    ;if (number < 0)
         BRGE    endIf
         LDWA    number,s    ;number = -number
         NEGA
         STWA    number,s
endIf:   @DECO   number,s    ;printf("%d", number)
         ADDSP   2,i         ;pop #number
         RET
```

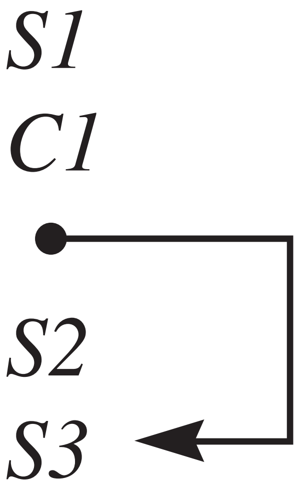
The load word instruction

- Instruction specifier: 1100 raaa
- Loads one word (two bytes) from memory to register r

$$r \leftarrow \mathrm{Oprnd} \; ; \; \boxed{N \leftarrow r < 0 \, , \, Z \leftarrow r = 0}$$

```
S1
if (C1) {
   S2
}
S3
```

(a) The structure at Level HOL6.



(b) The same structure at Level Asmb5.

```
         BR       main
number:  .EQUATE  0            ;local variable #2d
;
main:    SUBSP    2,i          ;push #number
         @DECI    number,s     ;scanf("%d", &number)
if:      LDWA     number,s     ;if (number < 0)
         BRGE     endIf
         LDWA     number,s     ;number = -number
         NEGA
         STWA     number,s
endIf:   @DECO    number,s     ;printf("%d", number)
         ADDSP    2,i          ;pop #number
         RET
```

Not necessary

Optimizing compiler

- Eliminates unnecessary instructions in the object code
- Advantage: Object code runs faster
- Disadvantage: Takes longer to compile

Compare word

- Instruction specifier: 1010 raaa
- Mnemonic: `CPWr` (`CPWA`, `CPWX`)
- Compare r with operand

$$\mathrm{T} \leftarrow \mathrm{r} - \mathrm{Oprnd}\ ;\ \mathrm{N} \leftarrow \mathrm{T} < 0\ ,\ \mathrm{Z} \leftarrow \mathrm{T} = 0\ ,$$

$$\mathrm{V} \leftarrow \{\mathit{overflow}\}\ ,\ \mathrm{C} \leftarrow \{\mathit{carry}\}\ ;\ \mathrm{N} \leftarrow \mathrm{N} \oplus \mathrm{V}$$

Figure 6.19

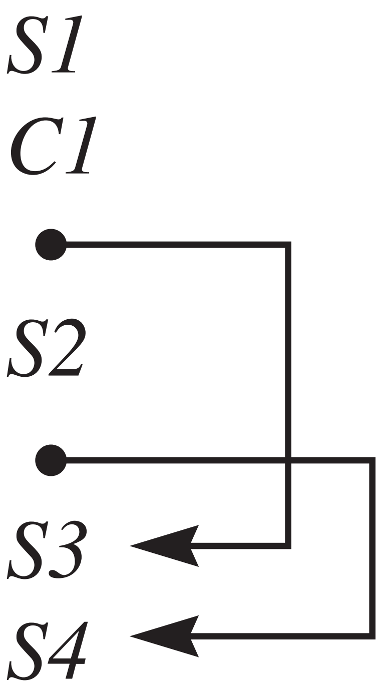
```
int main() {
    const int limit = 100;
    int num;
    scanf(”%d”, &num);
    if (num >= limit) {
        printf(”high\n”);
    } else {
        printf(”low\n”);
    }
    return 0;
}
```

Figure 6.19

```
         BR      main
limit:   .EQUATE 100         ;constant
num:     .EQUATE 0           ;local variable #2d
;
main:    SUBSP   2,i         ;push #num
         @DECI   num,s       ;scanf(”%d”, &num)
if:      LDWA    num,s       ;if (num >= limit)
         CPWA    limit,i
         BRLT    else
         @STRO   msg1,d      ;printf(”high\n”)
         BR      endIf
else:    @STRO   msg2,d      ;printf(”low\n”)
endIf:   ADDSP   2,i         ;pop #num
         RET
msg1:    .ASCII  ”high\n\0”
msg2:    .ASCII  ”low\n\0”
```

```
S1
if (C1) {
   S2
}
else {
   S3
}
S4
```

(a) The structure at Level HOL6.



(b) The same structure at Level Asmb5.

Figure 6.21

```
#include <stdio.h>

char letter;

int main() {
    scanf(”%c”, &letter);
    while (letter != ’*’) {
        if (letter == ’ ’) {
            printf(”\n”);
        } else {
            printf(”%c”, letter);
        }
        scanf(”%c”, &letter);
    }
    return 0;
}
```

Figure 6.21

```
#include <stdio.h>

char letter;

int main() {
    scanf(”%c”, &letter);
    while (letter != ’*’) {
        if (letter == ’ ’) {
            printf(”\n”);
        } else {
            printf(”%c”, letter);
        }
        scanf(”%c”, &letter);
    }
    return 0;
}
```

```
         BR      main
```

```
#include <stdio.h>

char letter;

int main() {
    scanf(”%c”, &letter);
    while (letter != ’*’) {
        if (letter == ’ ’) {
            printf(”\n”);
        } else {
            printf(”%c”, letter);
        }
        scanf(”%c”, &letter);
    }
    return 0;
}
```

```
         BR       main
letter:  .BLOCK   1           ;global variable #1c
```

Figure 6.21

```
#include <stdio.h>

char letter;

int main() {
    scanf("%c", &letter);
    while (letter != '*') {
        if (letter == ' ') {
            printf("\n");
        } else {
            printf("%c", letter);
        }
        scanf("%c", &letter);
    }
    return 0;
}
```

```
         BR      main
letter:  .BLOCK  1            ;global variable #1c
;
main:    @CHARI  letter,d     ;scanf("%c", &letter)
```

Figure 6.21

```
#include <stdio.h>

char letter;

int main() {
    scanf(”%c”, &letter);
    while (letter != ’*’) {
        if (letter == ’ ’) {
            printf(”\n”);
        } else {
            printf(”%c”, letter);
        }
        scanf(”%c”, &letter);
    }
    return 0;
}
```

```
         BR      main
letter:  .BLOCK  1           ;global variable #1c
;
main:    @CHARI  letter,d    ;scanf(”%c”, &letter)
while:   LDBA    letter,d    ;while (letter != ’*’)
```

Figure 6.21

```
#include <stdio.h>

char letter;

int main() {
    scanf(”%c”, &letter);
    while (letter != ’*’) {
        if (letter == ’ ’) {
            printf(”\n”);
        } else {
            printf(”%c”, letter);
        }
        scanf(”%c”, &letter);
    }
    return 0;
}
```

```
         BR      main
letter:  .BLOCK  1           ;global variable #1c
;
main:    @CHARI  letter,d    ;scanf(”%c”, &letter)
while:   LDBA    letter,d    ;while (letter != ’*’)
         CPBA    ’*’,i
```

Figure 6.21

```
#include <stdio.h>

char letter;

int main() {
    scanf(”%c”, &letter);
    while (letter != ’*’) {
        if (letter == ’ ’) {
            printf(”\n”);
        } else {
            printf(”%c”, letter);
        }
        scanf(”%c”, &letter);
    }
    return 0;
}
```

```
         BR      main
letter:  .BLOCK  1           ;global variable #1c
;
main:    @CHARI  letter,d    ;scanf(”%c”, &letter)
while:   LDBA    letter,d    ;while (letter != ’*’)
         CPBA    ’*’,i
         BREQ    endWh
```

Figure 6.21

```
#include <stdio.h>

char letter;

int main() {
    scanf(”%c”, &letter);
    while (letter != ’*’) {
        if (letter == ’ ’) {
            printf(”\n”);
        } else {
            printf(”%c”, letter);
        }
        scanf(”%c”, &letter);
    }
    return 0;
}
```

```
         BR       main
letter:  .BLOCK   1            ;global variable #1c
;
main:    @CHARI   letter,d     ;scanf(”%c”, &letter)
while:   LDBA     letter,d     ;while (letter != ’*’)
         CPBA     ’*’,i
         BREQ     endWh
if:      CPBA     ’ ’,i        ;if (letter == ’ ’)
```

Figure 6.21

```
#include <stdio.h>

char letter;

int main() {
    scanf(”%c”, &letter);
    while (letter != ’*’) {
        if (letter == ’ ’) {
            printf(”\n”);
        } else {
            printf(”%c”, letter);
        }
        scanf(”%c”, &letter);
    }
    return 0;
}
```

```
         BR      main
letter:  .BLOCK  1            ;global variable #1c
;
main:    @CHARI  letter,d     ;scanf(”%c”, &letter)
while:   LDBA    letter,d     ;while (letter != ’*’)
         CPBA    ’*’,i
         BREQ    endWh
if:      CPBA    ’ ’,i        ;if (letter == ’ ’)
         BRNE    else
```

```
#include <stdio.h>

char letter;

int main() {
    scanf(”%c”, &letter);
    while (letter != ’*’) {
        if (letter == ’ ’) {
            printf(”\n”);
        } else {
            printf(”%c”, letter);
        }
        scanf(”%c”, &letter);
    }
    return 0;
}
```

```
         BR      main
letter:  .BLOCK  1           ;global variable #1c
;
main:    @CHARI  letter,d    ;scanf(”%c”, &letter)
while:   LDBA    letter,d    ;while (letter != ’*’)
         CPBA    ’*’,i
         BREQ    endWh
if:      CPBA    ’ ’,i       ;if (letter == ’ ’)
         BRNE    else
         @CHARO  ’\n’,i      ;printf(”\n”)
```

Figure 6.21

```
#include <stdio.h>

char letter;

int main() {
    scanf(”%c”, &letter);
    while (letter != ’*’) {
        if (letter == ’ ’) {
            printf(”\n”);
        } else {
            printf(”%c”, letter);
        }
        scanf(”%c”, &letter);
    }
    return 0;
}
```

```
         BR       main
letter:  .BLOCK   1            ;global variable #1c
;
main:    @CHARI   letter,d     ;scanf(”%c”, &letter)
while:   LDBA     letter,d     ;while (letter != ’*’)
         CPBA     ’*’,i
         BREQ     endWh
if:      CPBA     ’ ’,i        ;if (letter == ’ ’)
         BRNE     else
         @CHARO   ’\n’,i       ;printf(”\n”)
         BR       endIf
```

Figure 6.21

```
#include <stdio.h>

char letter;

int main() {
    scanf(”%c”, &letter);
    while (letter != ’*’) {
        if (letter == ’ ’) {
            printf(”\n”);
        } else {
            printf(”%c”, letter);
        }
        scanf(”%c”, &letter);
    }
    return 0;
}
```

```
         BR       main
letter:  .BLOCK   1           ;global variable #1c
;
main:    @CHARI   letter,d    ;scanf(”%c”, &letter)
while:   LDBA     letter,d    ;while (letter != ’*’)
         CPBA     ’*’,i
         BREQ     endWh
if:      CPBA     ’ ’,i       ;if (letter == ’ ’)
         BRNE     else
         @CHARO   ’\n’,i      ;printf(”\n”)
         BR       endIf
else:    @CHARO   letter,d    ;printf(”%c”, letter)
```

```
#include <stdio.h>

char letter;

int main() {
    scanf(”%c”, &letter);
    while (letter != ’*’) {
        if (letter == ’ ’) {
            printf(”\n”);
        } else {
            printf(”%c”, letter);
        }
        scanf(”%c”, &letter);
    }
    return 0;
}
```

```
         BR      main
letter:  .BLOCK  1           ;global variable #1c
;
main:    @CHARI  letter,d    ;scanf(”%c”, &letter)
while:   LDBA    letter,d    ;while (letter != ’*’)
         CPBA    ’*’,i
         BREQ    endWh
if:      CPBA    ’ ’,i       ;if (letter == ’ ’)
         BRNE    else
         @CHARO  ’\n’,i      ;printf(”\n”)
         BR      endIf
else:    @CHARO  letter,d    ;printf(”%c”, letter)
endIf:   @CHARI  letter,d    ;scanf(”%c”, &letter)
```

Figure 6.21

```
#include <stdio.h>

char letter;

int main() {
    scanf(”%c”, &letter);
    while (letter != ’*’) {
        if (letter == ’ ’) {
            printf(”\n”);
        } else {
            printf(”%c”, letter);
        }
        scanf(”%c”, &letter);
    }
    return 0;
}
```

```
         BR      main
letter:  .BLOCK  1            ;global variable #1c
;
main:    @CHARI  letter,d     ;scanf(”%c”, &letter)
while:   LDBA    letter,d     ;while (letter != ’*’)
         CPBA    ’*’,i
         BREQ    endWh
if:      CPBA    ’ ’,i        ;if (letter == ’ ’)
         BRNE    else
         @CHARO  ’\n’,i       ;printf(”\n”)
         BR      endIf
else:    @CHARO  letter,d     ;printf(”%c”, letter)
endIf:   @CHARI  letter,d     ;scanf(”%c”, &letter)
         BR      while
```

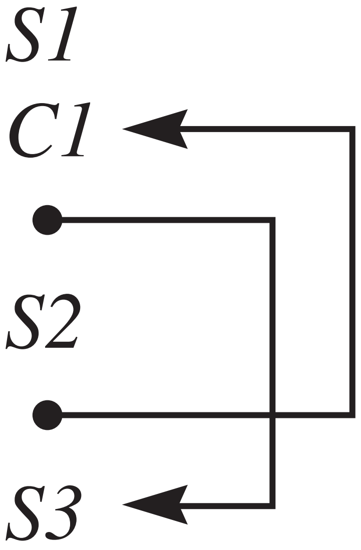
Figure 6.21

```
#include <stdio.h>

char letter;

int main() {
    scanf(”%c”, &letter);
    while (letter != ’*’) {
        if (letter == ’ ’) {
            printf(”\n”);
        } else {
            printf(”%c”, letter);
        }
        scanf(”%c”, &letter);
    }
    return 0;
}
```

```
         BR      main
letter:  .BLOCK  1           ;global variable #1c
;
main:    @CHARI  letter,d    ;scanf(”%c”, &letter)
while:   LDBA    letter,d    ;while (letter != ’*’)
         CPBA    ’*’,i
         BREQ    endWh
if:      CPBA    ’ ’,i       ;if (letter == ’ ’)
         BRNE    else
         @CHARO  ’\n’,i      ;printf(”\n”)
         BR      endIf
else:    @CHARO  letter,d    ;printf(”%c”, letter)
endIf:   @CHARI  letter,d    ;scanf(”%c”, &letter)
         BR      while
endWh:   RET
```

```
S1
while (C1) {
   S2
}
S3
```

(a) The structure at Level HOL6.



(b) The same structure at Level Asmb5.

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf(”%d”, cop);
    return 0;
}
```

Figure 6.23

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf(”%d”, cop);
    return 0;
}
```

```
         BR      main
```

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf("%d", cop);
    return 0;
}
```

```
         BR      main
cop:     .BLOCK  2             ;global variable #2d
```

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf("%d", cop);
    return 0;
}
```

```
         BR      main
cop:     .BLOCK  2           ;global variable #2d
driver:  .BLOCK  2           ;global variable #2d
```

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf("%d", cop);
    return 0;
}
```

```
         BR      main
cop:     .BLOCK  2            ;global variable #2d
driver:  .BLOCK  2            ;global variable #2d
;
main:    LDWA    0,i          ;cop = 0
```

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf("%d", cop);
    return 0;
}
```

```
         BR      main
cop:     .BLOCK  2            ;global variable #2d
driver:  .BLOCK  2            ;global variable #2d
;
main:    LDWA    0,i          ;cop = 0
         STWA    cop,d
```

Figure 6.23

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf("%d", cop);
    return 0;
}
```

```
         BR       main
cop:     .BLOCK   2            ;global variable #2d
driver:  .BLOCK   2            ;global variable #2d
;
main:    LDWA     0,i          ;cop = 0
         STWA     cop,d
         LDWA     40,i         ;driver = 40
```

Figure 6.23

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf(”%d”, cop);
    return 0;
}
```

```
         BR      main
cop:     .BLOCK  2            ;global variable #2d
driver:  .BLOCK  2            ;global variable #2d
;
main:    LDWA    0,i          ;cop = 0
         STWA    cop,d
         LDWA    40,i         ;driver = 40
         STWA    driver,d
```

Figure 6.23

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf(”%d”, cop);
    return 0;
}
```

```
         BR      main
cop:     .BLOCK  2           ;global variable #2d
driver:  .BLOCK  2           ;global variable #2d
;
main:    LDWA    0,i         ;cop = 0
         STWA    cop,d
         LDWA    40,i        ;driver = 40
         STWA    driver,d
do:      LDWA    cop,d       ;cop += 25
```

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf(”%d”, cop);
    return 0;
}
```

```
        BR      main
cop:    .BLOCK  2            ;global variable #2d
driver: .BLOCK  2            ;global variable #2d
;
main:   LDWA    0,i          ;cop = 0
        STWA    cop,d
        LDWA    40,i         ;driver = 40
        STWA    driver,d
do:     LDWA    cop,d        ;cop += 25
        ADDA    25,i
```

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf(”%d”, cop);
    return 0;
}
```

```
         BR      main
cop:     .BLOCK  2           ;global variable #2d
driver:  .BLOCK  2           ;global variable #2d
;
main:    LDWA    0,i         ;cop = 0
         STWA    cop,d
         LDWA    40,i        ;driver = 40
         STWA    driver,d
do:      LDWA    cop,d       ;cop += 25
         ADDA    25,i
         STWA    cop,d
```

Figure 6.23

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf(”%d”, cop);
    return 0;
}
```

```
         BR      main
cop:     .BLOCK  2           ;global variable #2d
driver:  .BLOCK  2           ;global variable #2d
;
main:    LDWA    0,i         ;cop = 0
         STWA    cop,d
         LDWA    40,i        ;driver = 40
         STWA    driver,d
do:      LDWA    cop,d       ;cop += 25
         ADDA    25,i
         STWA    cop,d
         LDWA    driver,d    ;driver += 20
```

Figure 6.23

```c
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf(”%d”, cop);
    return 0;
}
```

```
         BR      main
cop:     .BLOCK  2              ;global variable #2d
driver:  .BLOCK  2              ;global variable #2d
;
main:    LDWA    0,i            ;cop = 0
         STWA    cop,d
         LDWA    40,i           ;driver = 40
         STWA    driver,d
do:      LDWA    cop,d          ;cop += 25
         ADDA    25,i
         STWA    cop,d
         LDWA    driver,d       ;driver += 20
         ADDA    20,i
```

Figure 6.23

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf(”%d”, cop);
    return 0;
}
```

```
         BR      main
cop:     .BLOCK  2           ;global variable #2d
driver:  .BLOCK  2           ;global variable #2d
;
main:    LDWA    0,i         ;cop = 0
         STWA    cop,d
         LDWA    40,i        ;driver = 40
         STWA    driver,d
do:      LDWA    cop,d       ;cop += 25
         ADDA    25,i
         STWA    cop,d
         LDWA    driver,d    ;driver += 20
         ADDA    20,i
         STWA    driver,d
```

Figure 6.23

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf(”%d”, cop);
    return 0;
}
```

```
         BR       main
cop:     .BLOCK   2           ;global variable #2d
driver:  .BLOCK   2           ;global variable #2d
;
main:    LDWA     0,i         ;cop = 0
         STWA     cop,d
         LDWA     40,i        ;driver = 40
         STWA     driver,d
do:      LDWA     cop,d       ;cop += 25
         ADDA     25,i
         STWA     cop,d
         LDWA     driver,d    ;driver += 20
         ADDA     20,i
         STWA     driver,d
while:   LDWA     cop,d       ;while (cop < driver)
```

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf(”%d”, cop);
    return 0;
}
```

```
         BR      main
cop:     .BLOCK  2             ;global variable #2d
driver:  .BLOCK  2             ;global variable #2d
;
main:    LDWA    0,i           ;cop = 0
         STWA    cop,d
         LDWA    40,i          ;driver = 40
         STWA    driver,d
do:      LDWA    cop,d         ;cop += 25
         ADDA    25,i
         STWA    cop,d
         LDWA    driver,d      ;driver += 20
         ADDA    20,i
         STWA    driver,d
while:   LDWA    cop,d         ;while (cop < driver)
         CPWA    driver,d
```

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf(”%d”, cop);
    return 0;
}
```

```
         BR      main
cop:     .BLOCK  2             ;global variable #2d
driver:  .BLOCK  2             ;global variable #2d
;
main:    LDWA    0,i           ;cop = 0
         STWA    cop,d
         LDWA    40,i          ;driver = 40
         STWA    driver,d
do:      LDWA    cop,d         ;cop += 25
         ADDA    25,i
         STWA    cop,d
         LDWA    driver,d      ;driver += 20
         ADDA    20,i
         STWA    driver,d
while:   LDWA    cop,d         ;while (cop < driver)
         CPWA    driver,d
         BRLT    do
```

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf(”%d”, cop);
    return 0;
}
```

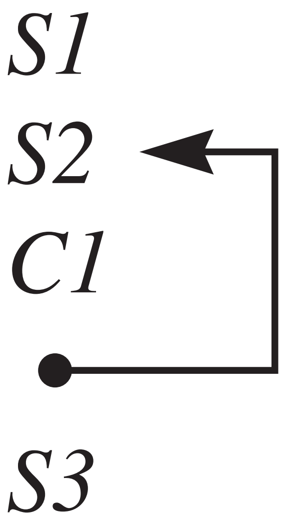
```
         BR       main
cop:     .BLOCK   2            ;global variable #2d
driver:  .BLOCK   2            ;global variable #2d
;
main:    LDWA     0,i          ;cop = 0
         STWA     cop,d
         LDWA     40,i         ;driver = 40
         STWA     driver,d
do:      LDWA     cop,d        ;cop += 25
         ADDA     25,i
         STWA     cop,d
         LDWA     driver,d     ;driver += 20
         ADDA     20,i
         STWA     driver,d
while:   LDWA     cop,d        ;while (cop < driver)
         CPWA     driver,d
         BRLT     do
         @DECO    cop,d        ;printf(”%d”, cop)
```

```
#include <stdio.h>

int cop;
int driver;

int main() {
    cop = 0;
    driver = 40;
    do {
        cop += 25;
        driver += 20;
    }
    while (cop < driver);
    printf(”%d”, cop);
    return 0;
}
```

```
         BR       main
cop:     .BLOCK   2            ;global variable #2d
driver:  .BLOCK   2            ;global variable #2d
;
main:    LDWA     0,i          ;cop = 0
         STWA     cop,d
         LDWA     40,i         ;driver = 40
         STWA     driver,d
do:      LDWA     cop,d        ;cop += 25
         ADDA     25,i
         STWA     cop,d
         LDWA     driver,d     ;driver += 20
         ADDA     20,i
         STWA     driver,d
while:   LDWA     cop,d        ;while (cop < driver)
         CPWA     driver,d
         BRLT     do
         @DECO    cop,d        ;printf(”%d”, cop)
         RET
```

```
S1
do {
    S2
}
while (C1)
S3
```

(**a**) The structure at Level HOL6.



(**b**) The same structure at Level Asmb5.

The `for` loop

- Initialize the control variable
- Test the control variable
- Execute the loop body
- Increment the control variable
- Branch to the test

```
#include <stdio.h>

int main() {
    int j;
    for (j = 0; j < 3; j++) {
        printf(”j = %d\n”, j);
    }
    return 0;
}
```

Figure 6.25

```
#include <stdio.h>

int main() {
    int j;
    for (j = 0; j < 3; j++) {
        printf(”j = %d\n”, j);
    }
    return 0;
}
```

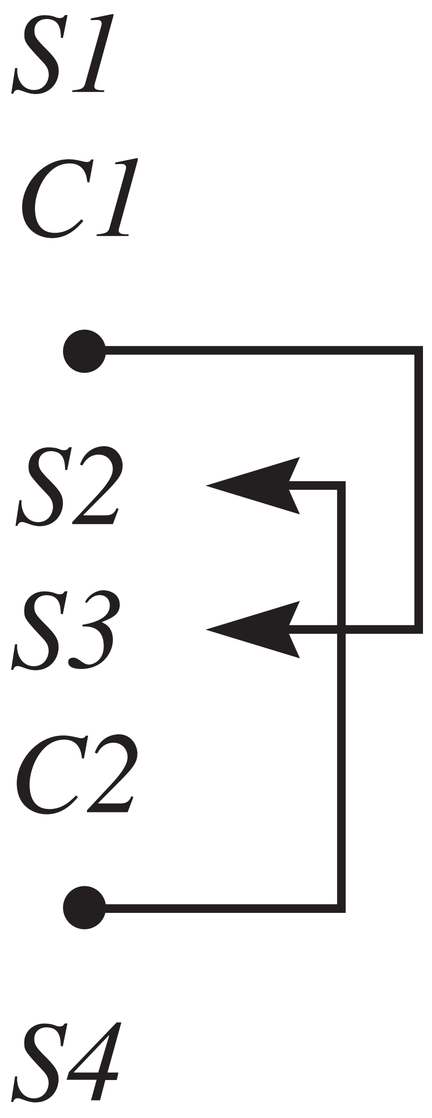
```
         BR      main
j:       .EQUATE 0          ;local variable #2d
;
main:    SUBSP   2,i        ;push #j
         LDWA    0,i        ;for (j = 0
         STWA    j,s
for:     CPWA    3,i        ;j < 3
         BRGE    endFor
         @STRO   msg,d      ;printf(”j = %d\n”, j)
         @DECO   j,s
         @CHARO  ’\n’,i
         LDWA    j,s        ;j++)
         ADDA    1,i
         STWA    j,s
         BR      for
endFor:  ADDSP   2,i        ;pop #j
         RET
msg:     .ASCII  ”j = \0”
```

Figure 6.14 Text (continued)

Assembly Language

```
0000   120003          BR       main
              j:       .EQUATE  0            ;local variable #2d
              ;
0003   580002 main:    SUBSP    2,i          ;push #j
0006   C00000          LDWA     0,i          ;for (j = 0
0009   E30000          STWA     j,s
000C   A00003 for:     CPWA     3,i          ;j < 3
000F   1C002A          BRGE     endFor
0012   49002E          STRO     msg,d        ;printf("j = %d\n", j)
0015   3B0000          DECO     j,s
0018   D0000A          LDBA     '\n',i
001B   F1FC16          STBA     charOut,d
001E   C30000          LDWA     j,s          ;j++)
0021   600001          ADDA     1,i
0024   E30000          STWA     j,s
0027   12000C          BR       for
002A   500002 endFor:  ADDSP    2,i          ;pop #j
002D   00              STOP
002E   6A203D msg:     .ASCII   "j = \x00"
       2000
0033                   .END
```

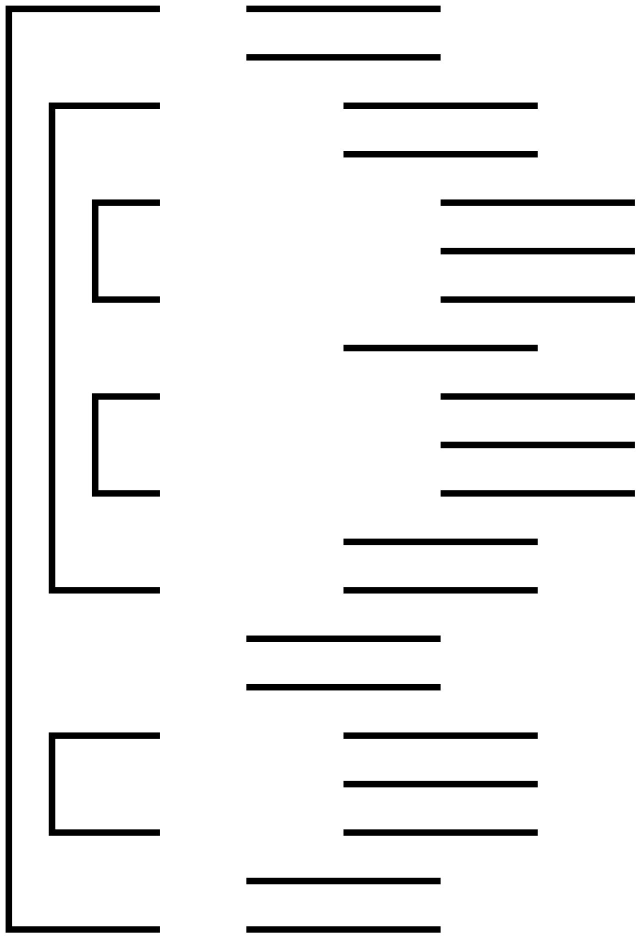
Not possible in many HOL6 languages



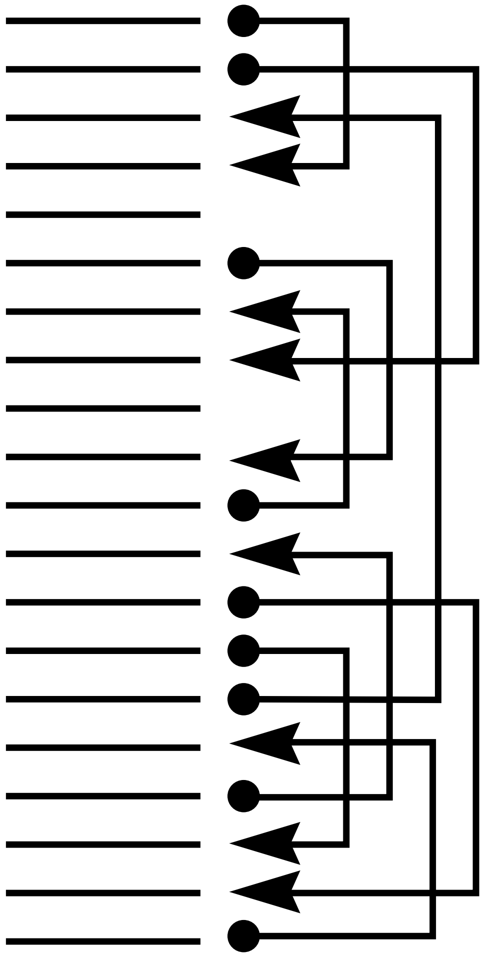
```
0000  120009          BR      main
0003  0000   n1:      .BLOCK  2           ;#2d
0005  0000   n2:      .BLOCK  2           ;#2d
0007  0000   n3:      .BLOCK  2           ;#2d
             ;
0009  310005 main:    DECI    n2,d
000C  310007          DECI    n3,d
000F  C10005          LDWA    n2,d
0012  A10007          CPWA    n3,d
0015  16002A          BRLT    L1
0018  310003          DECI    n1,d
001B  C10003          LDWA    n1,d
001E  A10007          CPWA    n3,d
0021  160074          BRLT    L7
0024  120065          BR      L6
0027  E10007          STWA    n3,d
002A  310003 L1:      DECI    n1,d
002D  C10005          LDWA    n2,d
0030  A10003          CPWA    n1,d
0033  160053          BRLT    L5
0036  390003          DECO    n1,d
0039  390005          DECO    n2,d
003C  390007 L2:      DECO    n3,d
003F  00              STOP
```

Figure 6.16 Text (continued)

```
0040   390005 L3:        DECO     n2,d
0043   390007            DECO     n3,d
0046   120081            BR       L9
0049   390003 L4:        DECO     n1,d
004C   390005            DECO     n2,d
004F   00                STOP
0050   E10003            STWA     n1,d
0053   C10007 L5:        LDWA     n3,d
0056   A10003            CPWA     n1,d
0059   160040            BRLT     L3
005C   390005            DECO     n2,d
005F   390003            DECO     n1,d
0062   12003C            BR       L2
0065   390007 L6:        DECO     n3,d
0068   C10003            LDWA     n1,d
006B   A10005            CPWA     n2,d
006E   160049            BRLT     L4
0071   12007E            BR       L8
0074   390003 L7:        DECO     n1,d
0077   390007            DECO     n3,d
007A   390005            DECO     n2,d
007D   00                STOP
007E   390005 L8:        DECO     n2,d
0081   390003 L9:        DECO     n1,d
0084   00                STOP
0085                     .END
```



(a) Structured flow.



(b) Spaghetti code.

The Structured Programming Theorem

Any algorithm containing goto's, no matter how complicated or unstructured, can be written with only nested if statements and while loops.

Bohm and Jacopini, 1966

The goto controversy

More recently I discovered why the use of the goto statement has such disastrous effects, and I became convinced that the goto statement should be abolished from all “higher level” programming languages. ... The goto statement as it stands is just too primitive; it is too much an invitation to make a mess of one’s program.

Dijkstra, 1968

The call subroutine instruction

- Instruction specifier: 0011 011a
- Mnemonic: `CALL`
- Push return address onto run-time stack

$$\mathrm{SP} \leftarrow \mathrm{SP} - 2\ ;\ \mathrm{Mem}[\mathrm{SP}] \leftarrow \mathrm{PC}\ ;\ \mathrm{PC} \leftarrow \mathrm{Oprnd}$$

The return from subroutine instruction

- Instruction specifier: 0000 0001
- Mnemonic: `RET`
- Pop return address off of run-time stack

$$PC \leftarrow \text{Mem}[SP] \; ; \; SP \leftarrow SP + 2$$

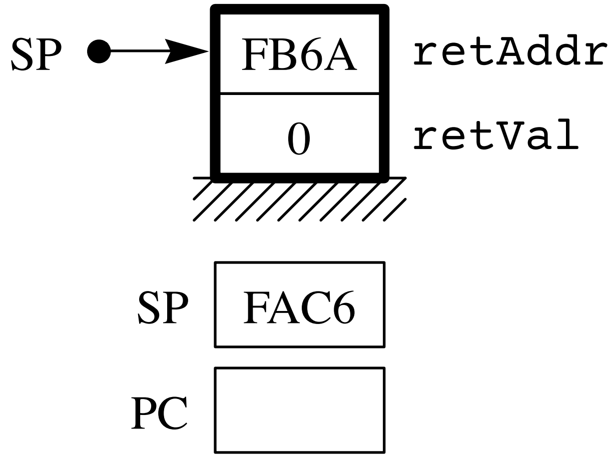
```
#include <stdio.h>
void printTri() {
    printf(”*\n”);
    printf(”**\n”);
    printf(”***\n”);
}
int main() {
    printTri();
    printTri();
    printTri();
    return 0;
}
```

Figure 6.27

```
#include <stdio.h>
void printTri() {
    printf(″*\n″);
    printf(″**\n″);
    printf(″***\n″);
}
int main() {
    printTri();
    printTri();
    printTri();
    return 0;
}
```

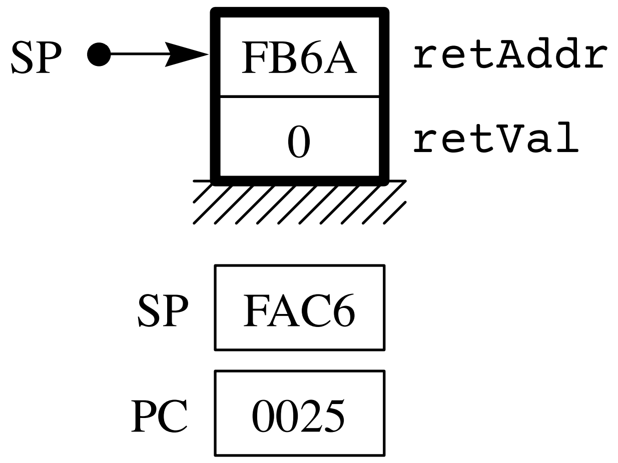
```
         BR      main
;
;******* void printTri()
printTri:@STRO   msg1,d       ;printf(″*\n″)
         @STRO   msg2,d       ;printf(″**\n″)
         @STRO   msg3,d       ;printf(″***\n″)
         RET
msg1:    .ASCII  ″*\n\0″
msg2:    .ASCII  ″**\n\0″
msg3:    .ASCII  ″***\n\0″
;
;******* int main()
main:    CALL    printTri     ;printTri()
         CALL    printTri     ;printTri()
         CALL    printTri     ;printTri()
         RET
```

```
...
             ;******* void printTri()
             ;printTri:@STRO   msg1,d      ;printf(”*\n”)
0003  C00003 printTri:LDWA    STRO,i
0006  390016          SCALL   msg1,d
             ;        End @STRO
...

0015  01              RET
...
             ;******* int main()
0022  360003 main:    CALL    printTri    ;printTri()
0025  360003          CALL    printTri    ;printTri()
...
```



(a) Before first `CALL`.

```
...
               ;******* void printTri()
               ;printTri:@STRO   msg1,d       ;printf("*\n")
0003   C00003 printTri:LDWA     STRO,i
0006   390016          SCALL    msg1,d
               ;        End @STRO
...

0015   01              RET
...
               ;******* int main()
0022   360003 main:    CALL     printTri      ;printTri()
0025   360003          CALL     printTri      ;printTri()
...
```

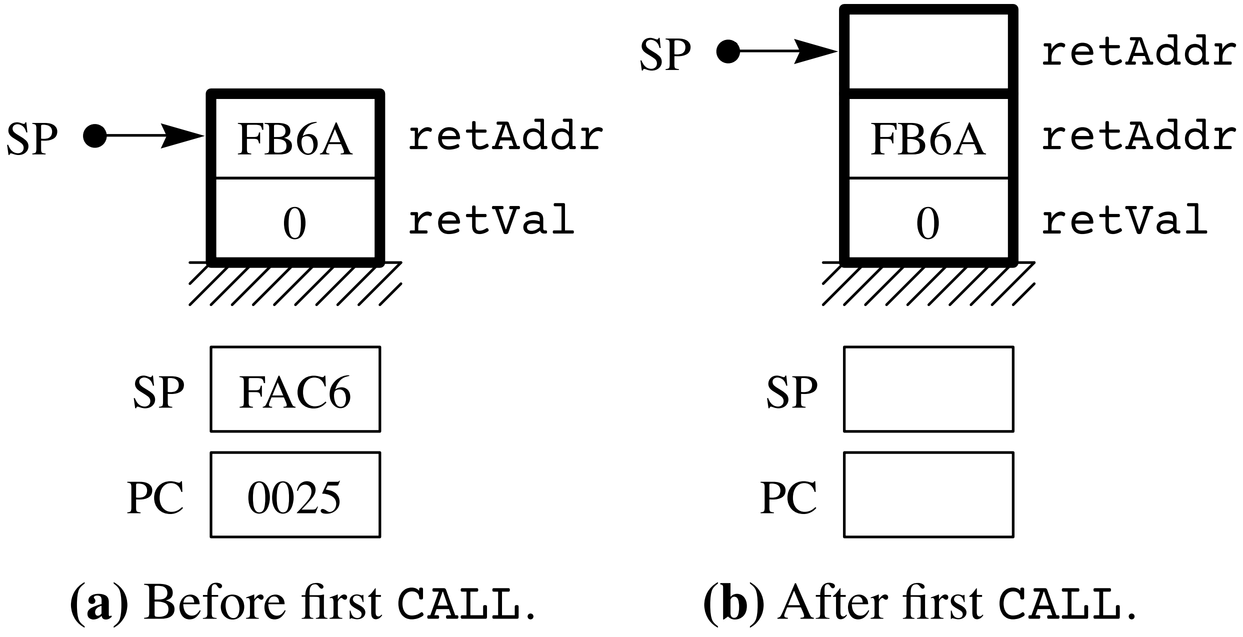


(a) Before first CALL.

```
...
             ;******* void printTri()
             ;printTri:@STRO   msg1,d      ;printf("*\n")
0003  C00003 printTri:LDWA    STRO,i
0006  390016          SCALL   msg1,d
             ;        End @STRO
...

0015  01              RET
...
             ;******* int main()
0022  360003 main:    CALL    printTri    ;printTri()
0025  360003          CALL    printTri    ;printTri()
...
```

Computer Systems Sixth edition

Figure 6.28

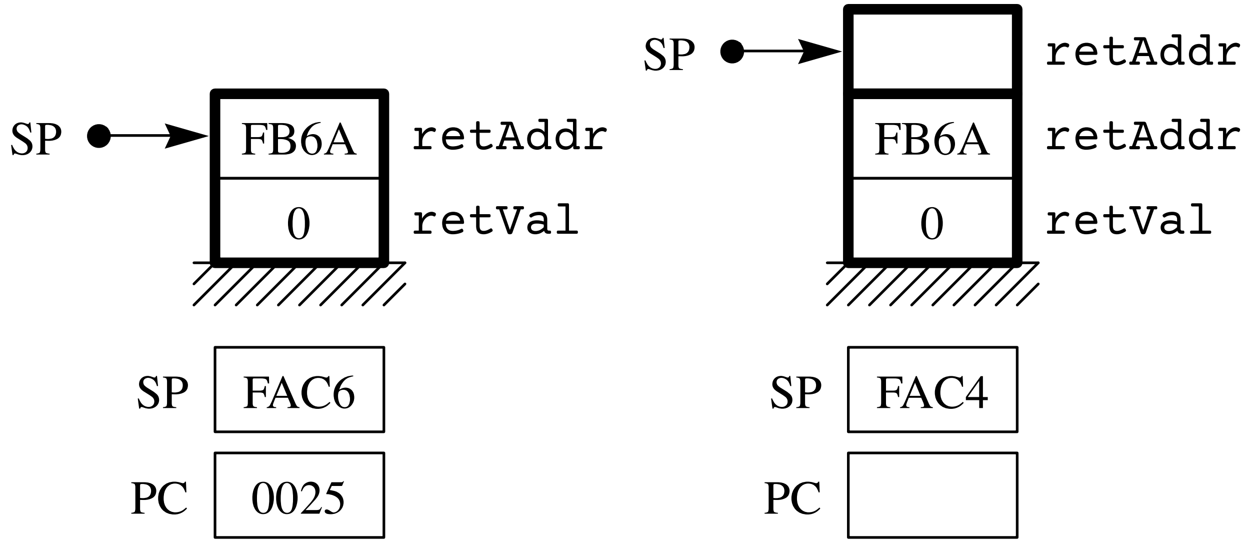


(a) Before first `CALL`. **(b)** After first `CALL`.

```
...
               ;******* void printTri()
               ;printTri:@STRO   msg1,d        ;printf("*\n")
0003   C00003 printTri:LDWA     STRO,i
0006   390016          SCALL    msg1,d
               ;        End @STRO
...

0015   01               RET
...
               ;******* int main()
0022   360003 main:     CALL     printTri      ;printTri()
0025   360003           CALL     printTri      ;printTri()
...
```

Computer Systems Sixth edition

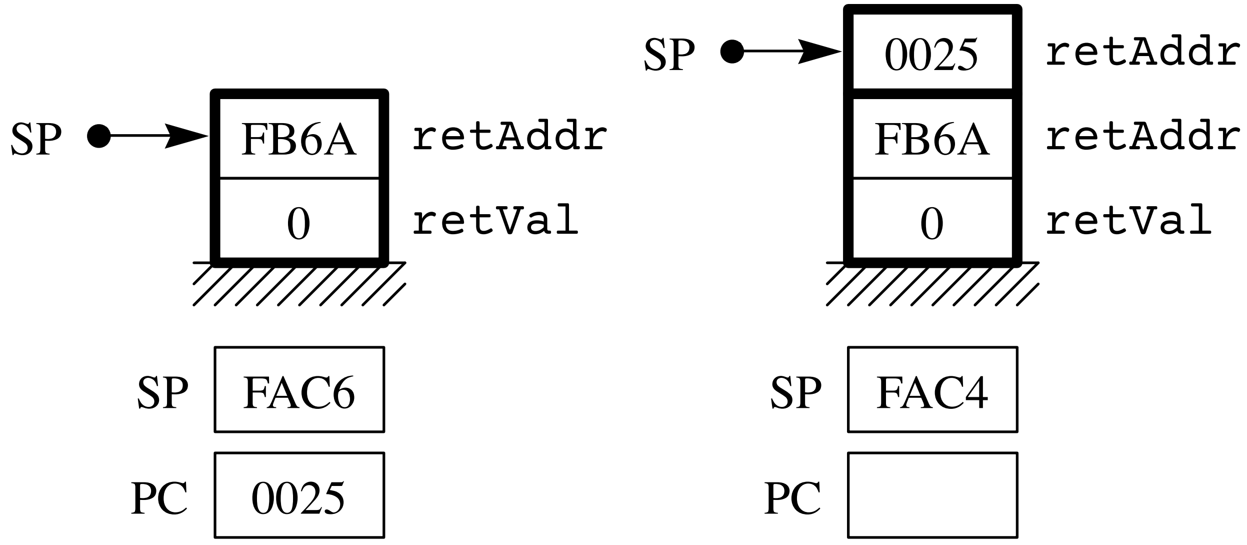
Figure 6.28



(a) Before first `CALL`.

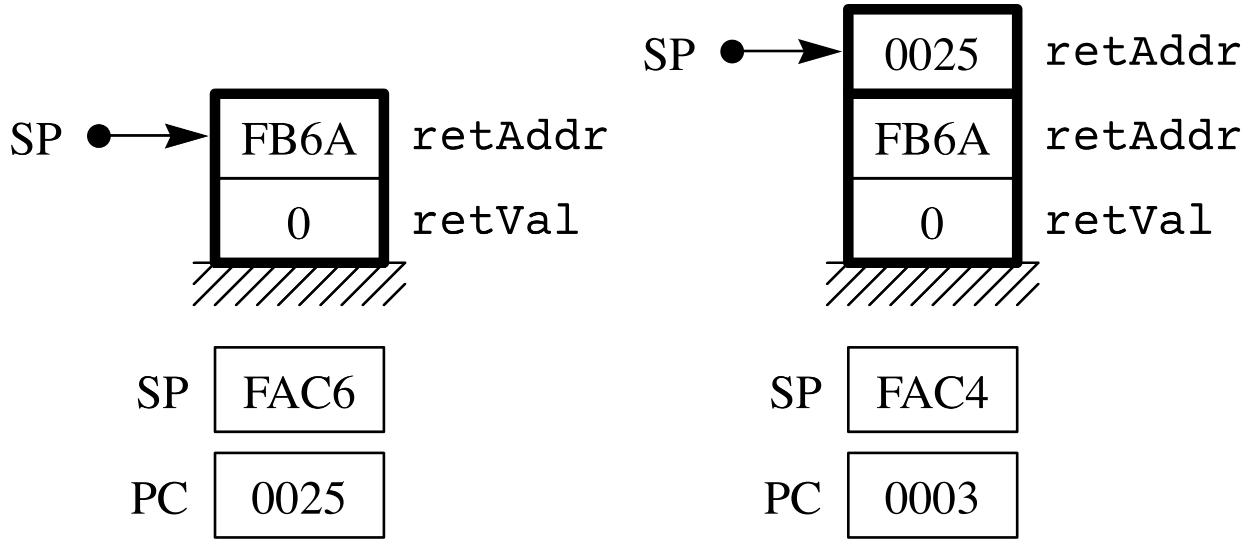
(b) After first `CALL`.

```
...
             ;******* void printTri()
             ;printTri:@STRO   msg1,d      ;printf(”*\n”)
0003  C00003 printTri:LDWA    STRO,i
0006  390016          SCALL   msg1,d
             ;        End @STRO
...

0015  01              RET
...
             ;******* int main()
0022  360003 main:    CALL    printTri    ;printTri()
0025  360003          CALL    printTri    ;printTri()
...
```



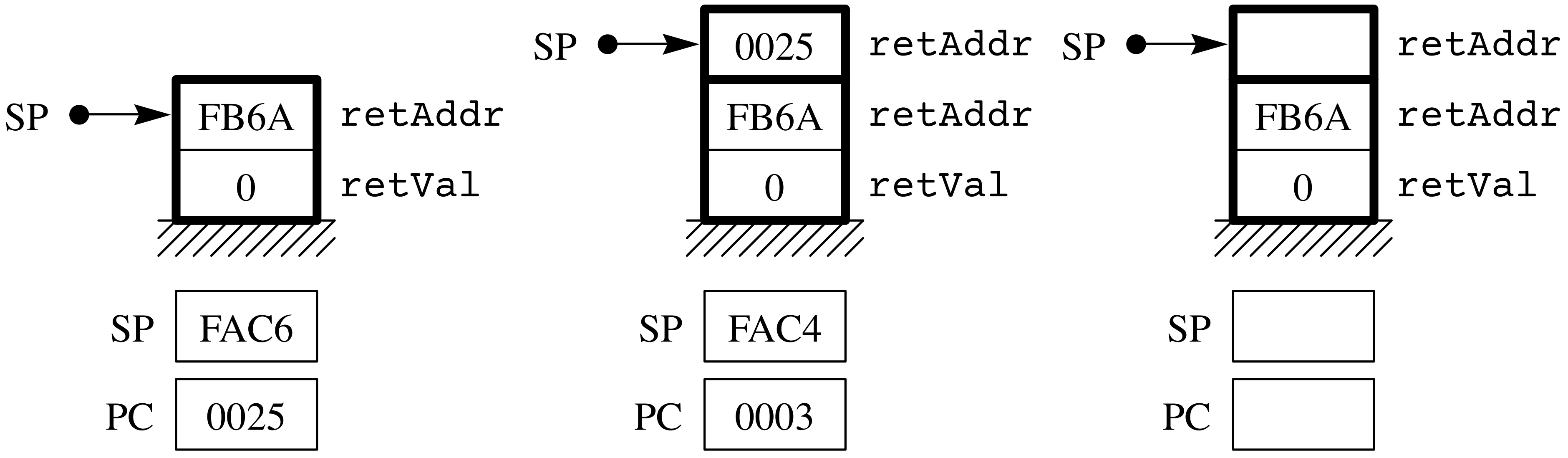
(a) Before first `CALL`. **(b)** After first `CALL`.

```
...
             ;******* void printTri()
             ;printTri:@STRO   msg1,d      ;printf(”*\n”)
0003  C00003 printTri:LDWA    STRO,i
0006  390016          SCALL   msg1,d
             ;        End @STRO
...

0015  01              RET
...
             ;******* int main()
0022  360003 main:    CALL    printTri    ;printTri()
0025  360003          CALL    printTri    ;printTri()
...
```



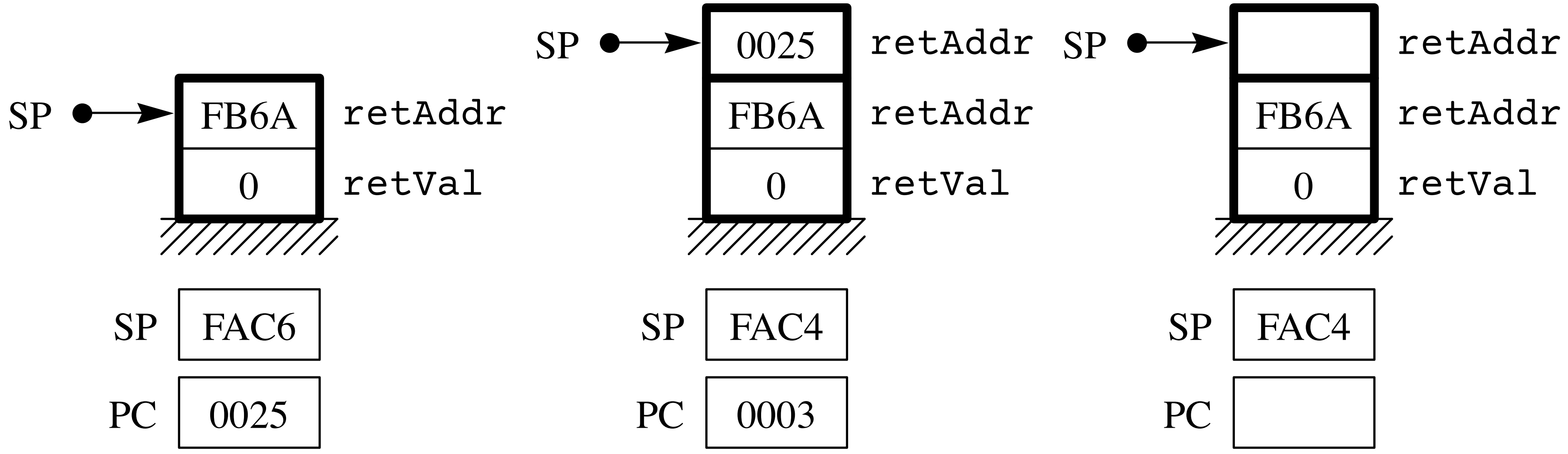
(a) Before first `CALL`. **(b)** After first `CALL`.

```
...
                ;******* void printTri()
                ;printTri:@STRO   msg1,d       ;printf("*\n")
0003   C00003 printTri:LDWA    STRO,i
0006   390016          SCALL   msg1,d
                ;        End @STRO
...

0015   01              RET
...
                ;******* int main()
0022   360003 main:    CALL    printTri       ;printTri()
0025   360003          CALL    printTri       ;printTri()
...
```



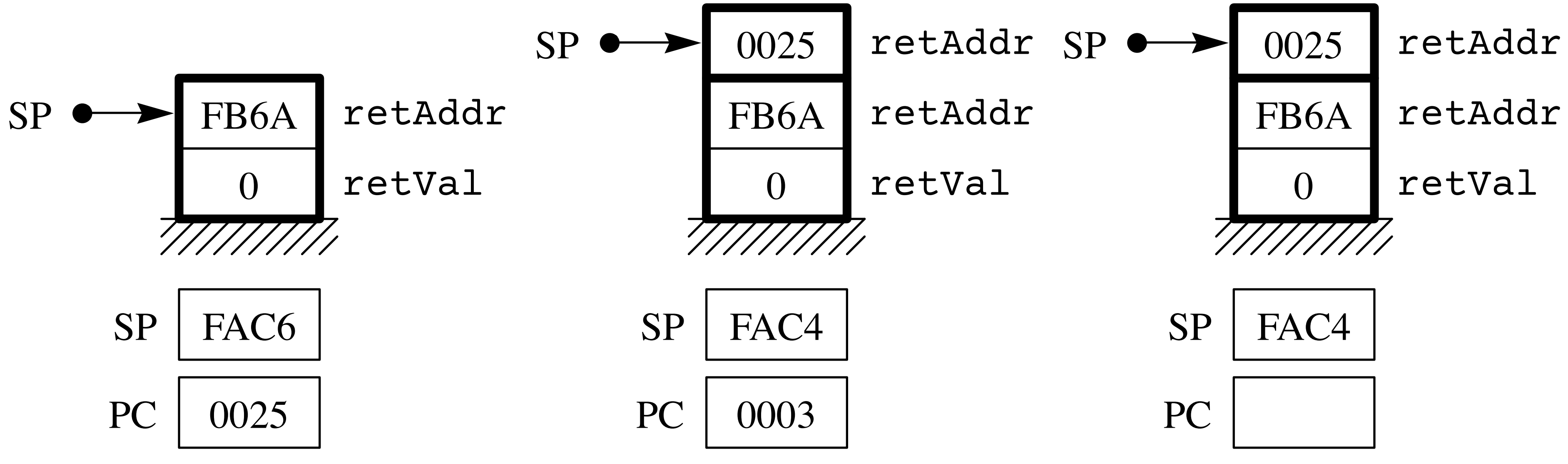
(a) Before first `CALL`. **(b)** After first `CALL`. **(c)** Before first `RET`.

```
...
             ;******* void printTri()
             ;printTri:@STRO   msg1,d      ;printf("*\n")
0003  C00003 printTri:LDWA    STRO,i
0006  390016          SCALL   msg1,d
             ;        End @STRO
...

0015  01              RET
...
             ;******* int main()
0022  360003 main:    CALL    printTri    ;printTri()
0025  360003          CALL    printTri    ;printTri()
...
```



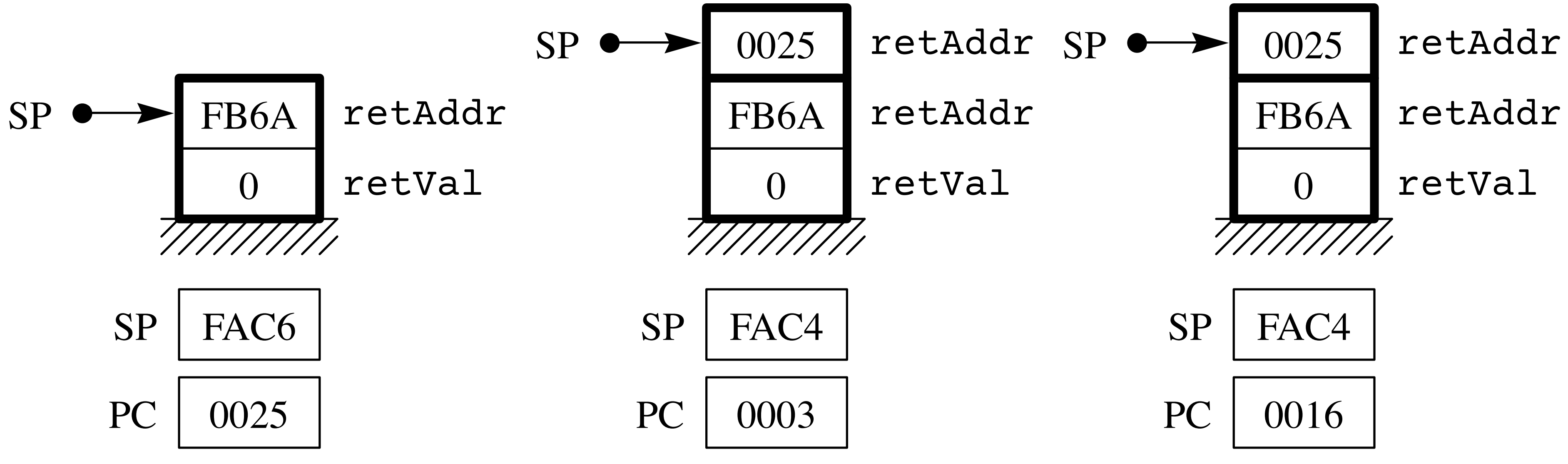
(a) Before first `CALL`. **(b)** After first `CALL`. **(c)** Before first `RET`.

```
...
               ;******* void printTri()
               ;printTri:@STRO   msg1,d       ;printf("*\n")
0003  C00003 printTri:LDWA    STRO,i
0006  390016          SCALL   msg1,d
               ;         End @STRO
...

0015  01               RET
...
               ;******* int main()
0022  360003 main:     CALL    printTri     ;printTri()
0025  360003           CALL    printTri     ;printTri()
...
```



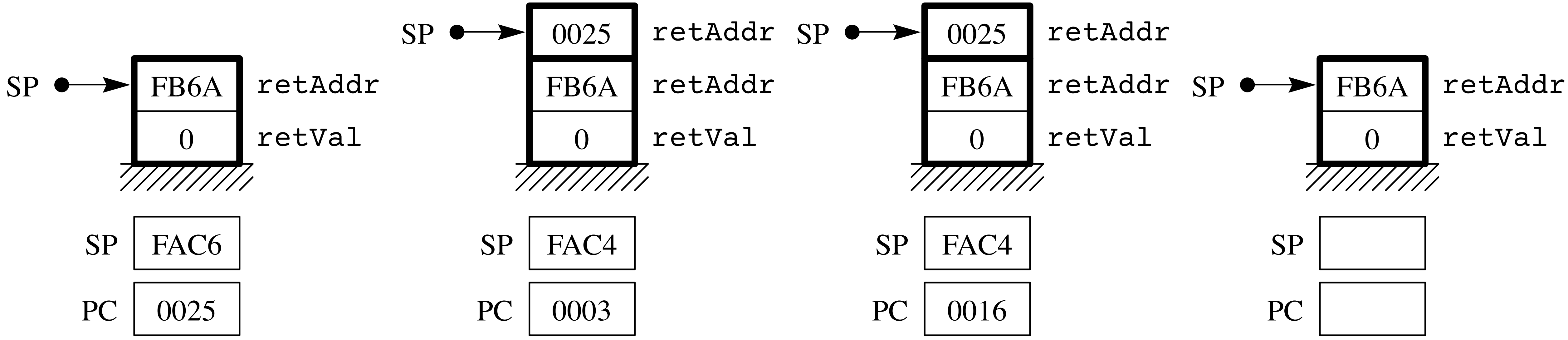
(a) Before first `CALL`. **(b)** After first `CALL`. **(c)** Before first `RET`.

```
...
             ;******* void printTri()
             ;printTri:@STRO   msg1,d       ;printf("*\n")
0003   C00003 printTri:LDWA     STRO,i
0006   390016          SCALL    msg1,d
             ;         End @STRO
...

0015   01              RET
...
             ;******* int main()
0022   360003 main:    CALL     printTri     ;printTri()
0025   360003          CALL     printTri     ;printTri()
...
```



(a) Before first `CALL`. **(b)** After first `CALL`. **(c)** Before first `RET`.

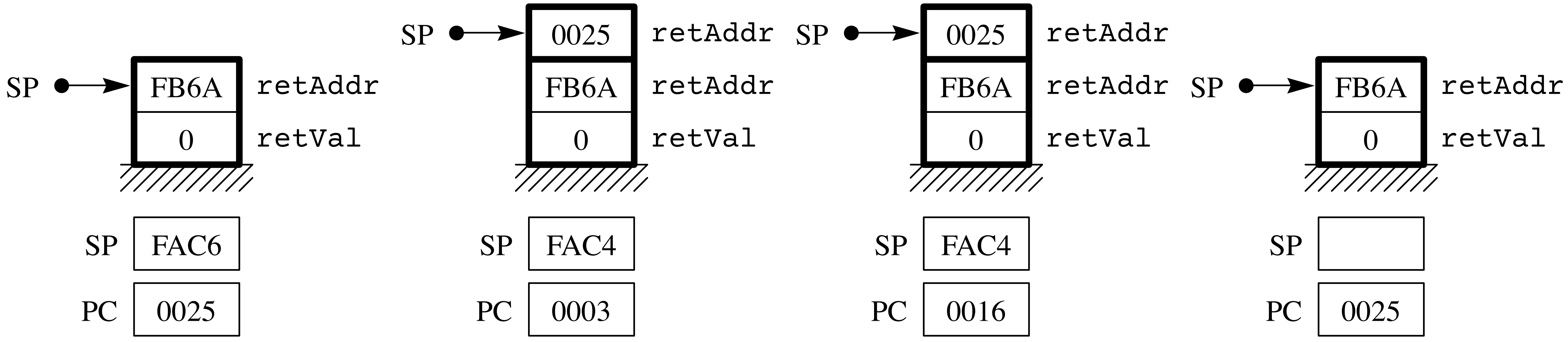
```
...
               ;******* void printTri()
               ;printTri:@STRO   msg1,d       ;printf("*\n")
0003   C00003 printTri:LDWA    STRO,i
0006   390016          SCALL   msg1,d
               ;       End @STRO
...

0015   01              RET
...
               ;******* int main()
0022   360003 main:    CALL    printTri     ;printTri()
0025   360003          CALL    printTri     ;printTri()
...
```



(a) Before first `CALL`. **(b)** After first `CALL`. **(c)** Before first `RET`. **(d)** After first `RET`.

```
...
              ;******* void printTri()
              ;printTri:@STRO   msg1,d      ;printf("*\n")
0003   C00003 printTri:LDWA    STRO,i
0006   390016          SCALL   msg1,d
              ;        End @STRO
...

0015   01              RET
...
              ;******* int main()
0022   360003 main:    CALL    printTri     ;printTri()
0025   360003          CALL    printTri     ;printTri()
...
```

Figure 6.28



(a) Before first `CALL`.

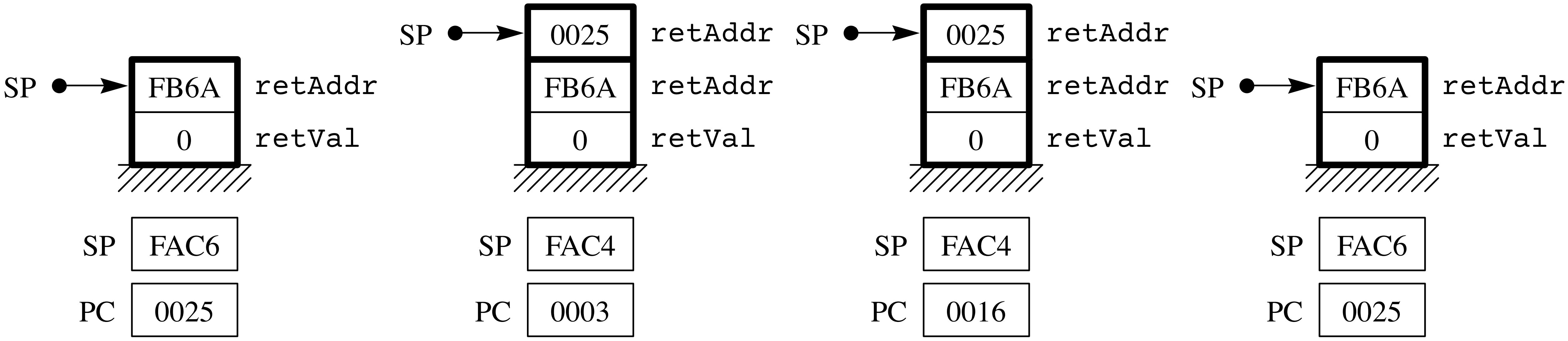
(b) After first `CALL`.

(c) Before first `RET`.

(d) After first `RET`.

```
...
              ;******* void printTri()
              ;printTri:@STRO   msg1,d      ;printf("*\n")
0003   C00003 printTri:LDWA     STRO,i
0006   390016          SCALL    msg1,d
              ;        End @STRO
...

0015   01              RET
...
              ;******* int main()
0022   360003 main:    CALL     printTri    ;printTri()
0025   360003          CALL     printTri    ;printTri()
...
```

Figure 6.28



(a) Before first `CALL`. **(b)** After first `CALL`. **(c)** Before first `RET`. **(d)** After first `RET`.

```
...
             ;******* void printTri()
             ;printTri:@STRO   msg1,d       ;printf("*\n")
0003   C00003 printTri:LDWA    STRO,i
0006   390016          SCALL   msg1,d
             ;         End @STRO
...

0015   01              RET
...
             ;******* int main()
0022   360003 main:    CALL    printTri     ;printTri()
0025   360003          CALL    printTri     ;printTri()
...
```

To call a `void` function in C

- Push the actual parameters
- Push the return address
- Push storage for the local variables

In assembly language

In assembly language

- Caller pushes actual parameters (executes `SUBSP`)

In assembly language

- Caller pushes actual parameters (executes `SUBSP`)
- Caller pushes return address (executes `CALL`)

In assembly language

- Caller pushes actual parameters (executes `SUBSP`)
- Caller pushes return address (executes `CALL`)
- Callee allocates local variables (executes `SUBSP`)

In assembly language

- Caller pushes actual parameters (executes `SUBSP`)
- Caller pushes return address (executes `CALL`)
- Callee allocates local variables (executes `SUBSP`)
- Callee executes its body.

In assembly language

- Caller pushes actual parameters (executes `SUBSP`)
- Caller pushes return address (executes `CALL`)
- Callee allocates local variables (executes `SUBSP`)
- Callee executes its body.
- Callee pops local variables (executes `ADDSP`)

In assembly language

- Caller pushes actual parameters (executes `SUBSP`)
- Caller pushes return address (executes `CALL`)
- Callee allocates local variables (executes `SUBSP`)
- Callee executes its body.
- Callee pops local variables (executes `ADDSP`)
- Callee pops return address (executes `RET`)

In assembly language

- Caller pushes actual parameters (executes `SUBSP`)
- Caller pushes return address (executes `CALL`)
- Callee allocates local variables (executes `SUBSP`)
- Callee executes its body.
- Callee pops local variables (executes `ADDSP`)
- Callee pops return address (executes `RET`)
- Caller pops actual parameters (executes `ADDSP`)

Function call with global variables

- To access the actual parameter in the caller, use direct addressing (`d`)
- To access the formal parameter in the callee, use stack-relative addressing (`s`)

Figure 6.29

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
```

Code for `printBar()`

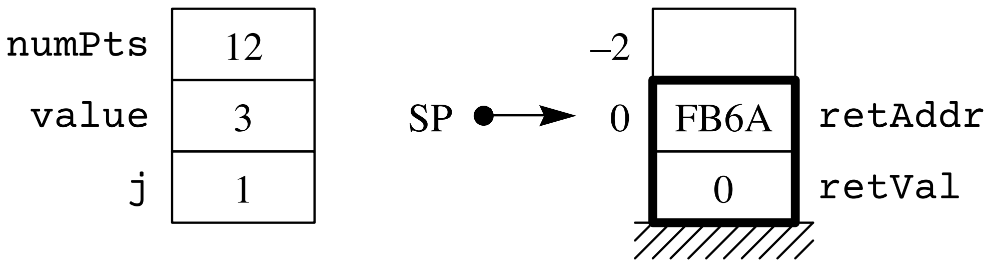
```
;******* main()
```

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
```

Code for `printBar( )`

```
;******* main()
```

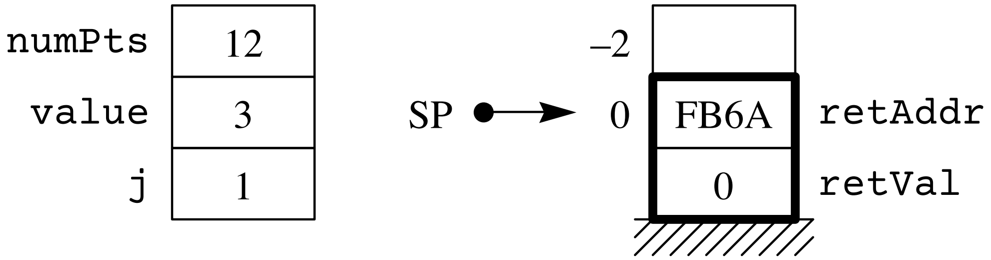


```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR       main
numPts:  .BLOCK   2            ;global variable #2d
```

Code for `printBar( )`

```
;******* main()
```

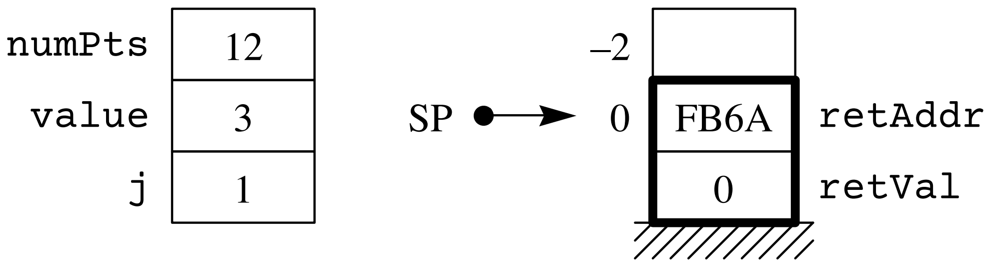


```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR       main
numPts:  .BLOCK   2             ;global variable #2d
value:   .BLOCK   2             ;global variable #2d
```

Code for `printBar()`

```
;******* main()
```

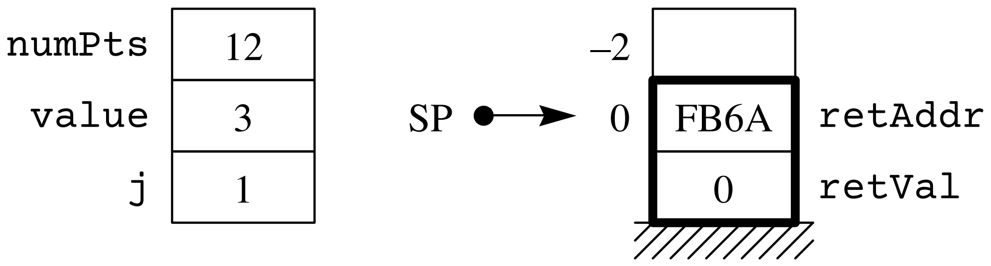


```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2             ;global variable #2d
value:   .BLOCK  2             ;global variable #2d
j:       .BLOCK  2             ;global variable #2d
```

Code for `printBar()`

```
;******* main()
```



```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2             ;global variable #2d
value:   .BLOCK  2             ;global variable #2d
j:       .BLOCK  2             ;global variable #2d
```

Code for `printBar()`

```
;******* main()
main:    @DECI   numPts,d    ;scanf(”%d”, &numPts)
```

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
          BR        main
numPts:   .BLOCK    2             ;global variable #2d
value:    .BLOCK    2             ;global variable #2d
j:        .BLOCK    2             ;global variable #2d
```

Code for `printBar()`

```
;******* main()
main:    @DECI    numPts,d     ;scanf(”%d”, &numPts)
         LDWA     1,i          ;for (j = 1
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
          BR      main
numPts:   .BLOCK  2            ;global variable #2d
value:    .BLOCK  2            ;global variable #2d
j:        .BLOCK  2            ;global variable #2d
```

Code for `printBar()`

```
;******* main()
main:    @DECI    numPts,d     ;scanf(”%d”, &numPts)
         LDWA     1,i          ;for (j = 1
         STWA     j,d
```

Figure 6.29

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
          BR       main
numPts:   .BLOCK   2             ;global variable #2d
value:    .BLOCK   2             ;global variable #2d
j:        .BLOCK   2             ;global variable #2d
```

Code for `printBar( )`

```
;******* main()
main:    @DECI   numPts,d      ;scanf(”%d”, &numPts)
         LDWA    1,i           ;for (j = 1
         STWA    j,d
for2:    CPWA    numPts,d      ;j <= numPts
```

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2            ;global variable #2d
value:   .BLOCK  2            ;global variable #2d
j:       .BLOCK  2            ;global variable #2d
```

Code for `printBar( )`

```
;******* main()
main:    @DECI   numPts,d     ;scanf("%d", &numPts)
         LDWA    1,i          ;for (j = 1
         STWA    j,d
for2:    CPWA    numPts,d     ;j <= numPts
         BRGT    endFor2
```

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2            ;global variable #2d
value:   .BLOCK  2            ;global variable #2d
j:       .BLOCK  2            ;global variable #2d
```

Code for `printBar()`

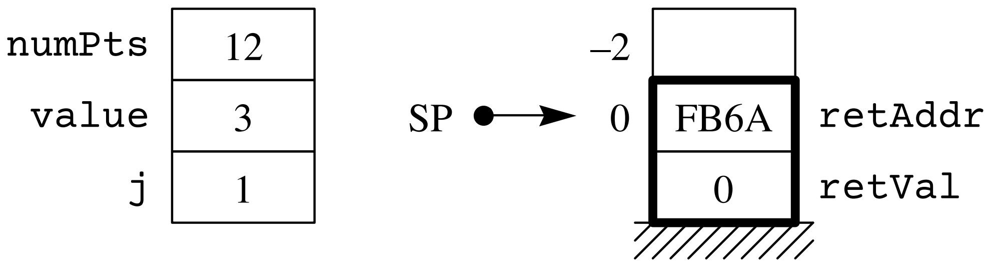
```
;******* main()
main:    @DECI   numPts,d     ;scanf(”%d”, &numPts)
         LDWA    1,i          ;for (j = 1
         STWA    j,d
for2:    CPWA    numPts,d     ;j <= numPts
         BRGT    endFor2
         @DECI   value,d      ;scanf(”%d”, &value)
```

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2            ;global variable #2d
value:   .BLOCK  2            ;global variable #2d
j:       .BLOCK  2            ;global variable #2d
```

Code for `printBar()`

```
;******* main()
main:    @DECI   numPts,d     ;scanf("%d", &numPts)
         LDWA    1,i          ;for (j = 1
         STWA    j,d
for2:    CPWA    numPts,d     ;j <= numPts
         BRGT    endFor2
         @DECI   value,d      ;scanf("%d", &value)
```

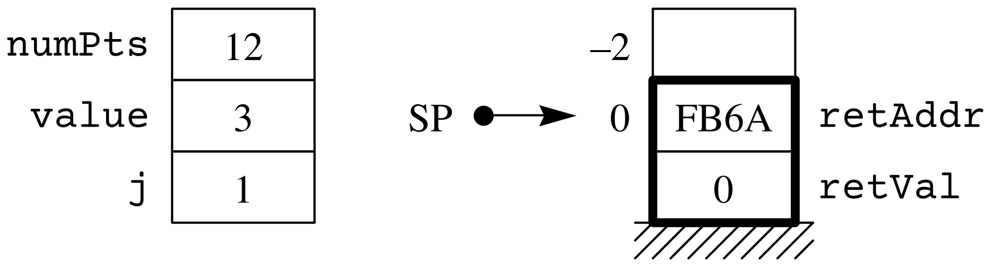


```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR       main
numPts:  .BLOCK   2             ;global variable #2d
value:   .BLOCK   2             ;global variable #2d
j:       .BLOCK   2             ;global variable #2d
```

Code for `printBar( )`

```
;******* main()
main:    @DECI    numPts,d      ;scanf("%d", &numPts)
         LDWA     1,i           ;for (j = 1
         STWA     j,d
for2:    CPWA     numPts,d      ;j <= numPts
         BRGT     endFor2
         @DECI    value,d       ;scanf("%d", &value)
         LDWA     value,d       ;move value
```

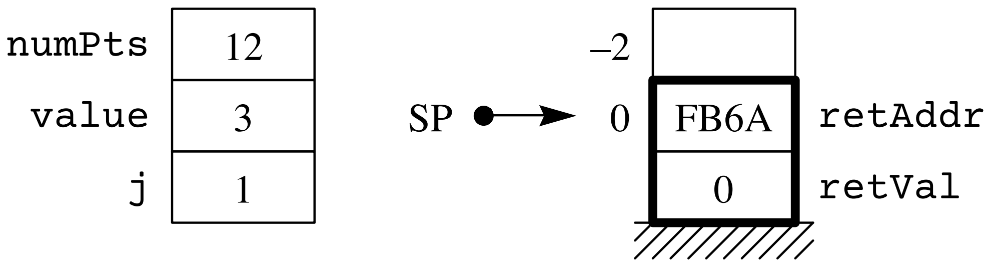


```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2            ;global variable #2d
value:   .BLOCK  2            ;global variable #2d
j:       .BLOCK  2            ;global variable #2d
```

Code for `printBar()`

```
;******* main()
main:    @DECI   numPts,d     ;scanf("%d", &numPts)
         LDWA    1,i          ;for (j = 1
         STWA    j,d
for2:    CPWA    numPts,d     ;j <= numPts
         BRGT    endFor2
         @DECI   value,d      ;scanf("%d", &value)
         LDWA    value,d      ;move value
         STWA    -2,s
```

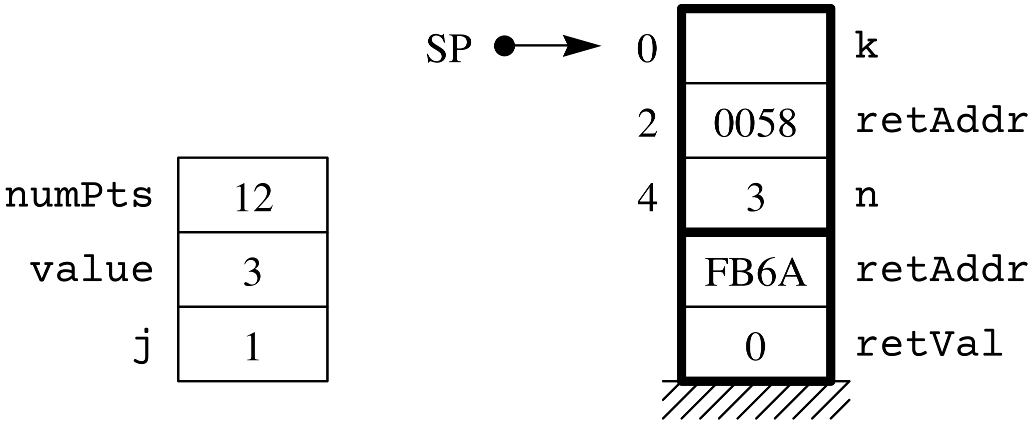


```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR        main
numPts:  .BLOCK    2           ;global variable #2d
value:   .BLOCK    2           ;global variable #2d
j:       .BLOCK    2           ;global variable #2d
```

Code for printBar()

```
;******* main()
main:    @DECI    numPts,d     ;scanf("%d", &numPts)
         LDWA     1,i          ;for (j = 1
         STWA     j,d
for2:    CPWA     numPts,d     ;j <= numPts
         BRGT     endFor2
         @DECI    value,d      ;scanf("%d", &value)
         LDWA     value,d      ;move value
         STWA     -2,s
```

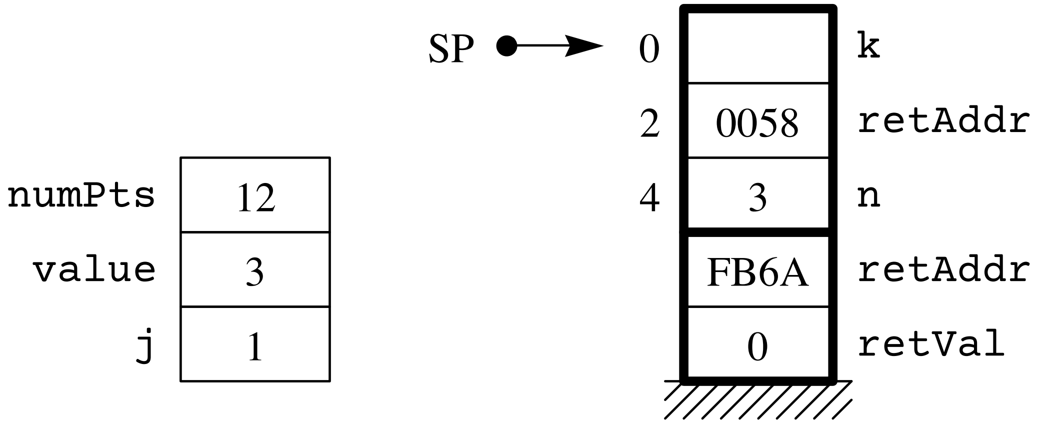


```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2            ;global variable #2d
value:   .BLOCK  2            ;global variable #2d
j:       .BLOCK  2            ;global variable #2d
```

Code for `printBar()`

```
;******* main()
main:    @DECI   numPts,d     ;scanf("%d", &numPts)
         LDWA    1,i          ;for (j = 1
         STWA    j,d
for2:    CPWA    numPts,d     ;j <= numPts
         BRGT    endFor2
         @DECI   value,d      ;scanf("%d", &value)
         LDWA    value,d      ;move value
         STWA    -2,s
         SUBSP   2,i          ;push #n
```

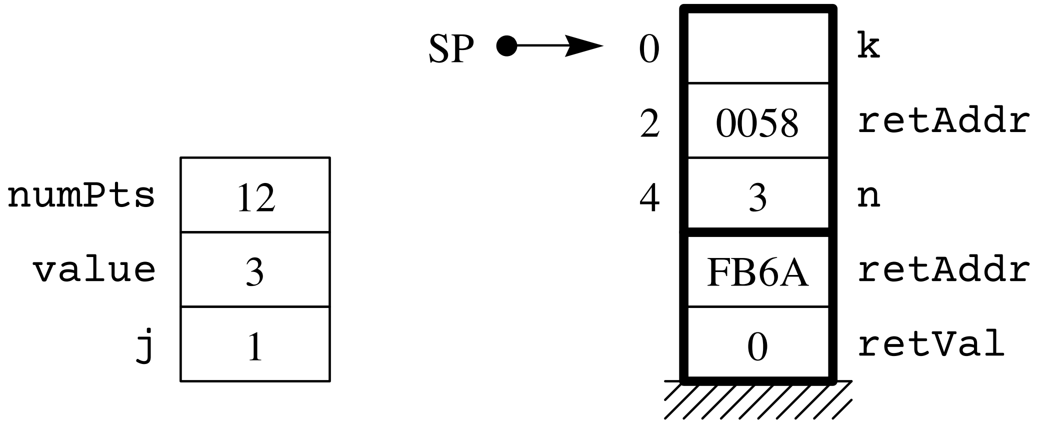


```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2             ;global variable #2d
value:   .BLOCK  2             ;global variable #2d
j:       .BLOCK  2             ;global variable #2d
```

Code for printBar( )

```
;******* main()
main:    @DECI   numPts,d      ;scanf("%d", &numPts)
         LDWA    1,i           ;for (j = 1
         STWA    j,d
for2:    CPWA    numPts,d      ;j <= numPts
         BRGT    endFor2
         @DECI   value,d       ;scanf("%d", &value)
         LDWA    value,d       ;move value
         STWA    -2,s
         SUBSP   2,i           ;push #n
         CALL    printBar      ;printBar(value)
```



```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR        main
numPts:  .BLOCK    2             ;global variable #2d
value:   .BLOCK    2             ;global variable #2d
j:       .BLOCK    2             ;global variable #2d
```

Code for printBar()

```
;******* main()
main:    @DECI     numPts,d      ;scanf("%d", &numPts)
         LDWA      1,i           ;for (j = 1
         STWA      j,d
for2:    CPWA      numPts,d      ;j <= numPts
         BRGT      endFor2
         @DECI     value,d       ;scanf("%d", &value)
         LDWA      value,d       ;move value
         STWA      -2,s
         SUBSP     2,i           ;push #n
         CALL      printBar      ;printBar(value)
         ADDSP     2,i           ;pop #n
```

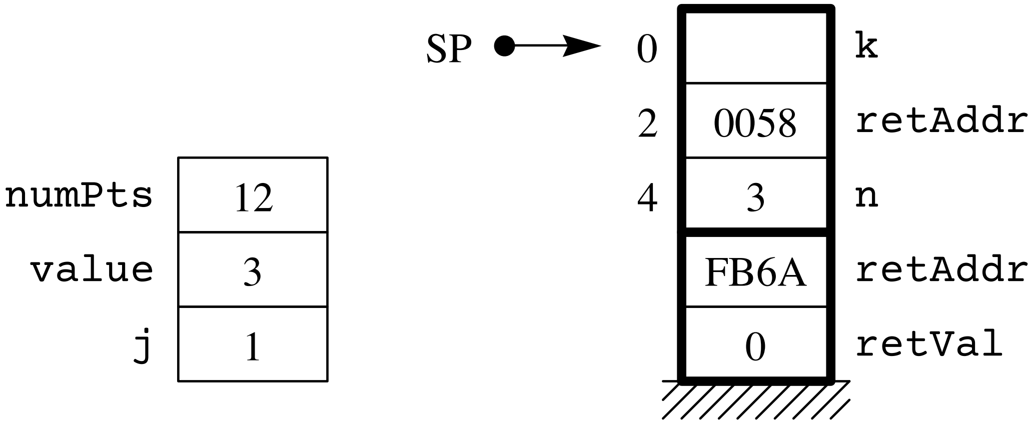


Figure 6.29

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(″*″);
    }
    printf(″\n″);
}

int main() {
    scanf(″%d″, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(″%d″, &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2            ;global variable #2d
value:   .BLOCK  2            ;global variable #2d
j:       .BLOCK  2            ;global variable #2d
```

Code for `printBar()`

```
;******* main()
main:    @DECI   numPts,d     ;scanf(″%d″, &numPts)
         LDWA    1,i          ;for (j = 1
         STWA    j,d
for2:    CPWA    numPts,d     ;j <= numPts
         BRGT    endFor2
         @DECI   value,d      ;scanf(″%d″, &value)
         LDWA    value,d      ;move value
         STWA    -2,s
         SUBSP   2,i          ;push #n
         CALL    printBar     ;printBar(value)
         ADDSP   2,i          ;pop #n
         LDWA    j,d          ;j++)
```

Figure 6.29

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
          BR       main
numPts:   .BLOCK   2             ;global variable #2d
value:    .BLOCK   2             ;global variable #2d
j:        .BLOCK   2             ;global variable #2d
```

Code for `printBar( )`

```
;******* main()
main:    @DECI    numPts,d     ;scanf(”%d”, &numPts)
         LDWA     1,i          ;for (j = 1
         STWA     j,d
for2:    CPWA     numPts,d     ;j <= numPts
         BRGT     endFor2
         @DECI    value,d      ;scanf(”%d”, &value)
         LDWA     value,d      ;move value
         STWA     -2,s
         SUBSP    2,i          ;push #n
         CALL     printBar     ;printBar(value)
         ADDSP    2,i          ;pop #n
         LDWA     j,d          ;j++)
         ADDA     1,i
```

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR       main
numPts:  .BLOCK   2             ;global variable #2d
value:   .BLOCK   2             ;global variable #2d
j:       .BLOCK   2             ;global variable #2d
```

Code for `printBar()`

```
;******* main()
main:    @DECI    numPts,d      ;scanf("%d", &numPts)
         LDWA     1,i           ;for (j = 1
         STWA     j,d
for2:    CPWA     numPts,d      ;j <= numPts
         BRGT     endFor2
         @DECI    value,d       ;scanf("%d", &value)
         LDWA     value,d       ;move value
         STWA     -2,s
         SUBSP    2,i           ;push #n
         CALL     printBar      ;printBar(value)
         ADDSP    2,i           ;pop #n
         LDWA     j,d           ;j++)
         ADDA     1,i
         STWA     j,d
```

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2             ;global variable #2d
value:   .BLOCK  2             ;global variable #2d
j:       .BLOCK  2             ;global variable #2d
```

Code for `printBar()`

```
;******* main()
main:    @DECI   numPts,d      ;scanf(”%d”, &numPts)
         LDWA    1,i           ;for (j = 1
         STWA    j,d
for2:    CPWA    numPts,d      ;j <= numPts
         BRGT    endFor2
         @DECI   value,d       ;scanf(”%d”, &value)
         LDWA    value,d       ;move value
         STWA    -2,s
         SUBSP   2,i           ;push #n
         CALL    printBar      ;printBar(value)
         ADDSP   2,i           ;pop #n
         LDWA    j,d           ;j++)
         ADDA    1,i
         STWA    j,d
         BR      for2
```

Figure 6.29

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR       main
numPts:  .BLOCK   2             ;global variable #2d
value:   .BLOCK   2             ;global variable #2d
j:       .BLOCK   2             ;global variable #2d
```

Code for `printBar( )`

```
;******* main()
main:    @DECI    numPts,d      ;scanf(”%d”, &numPts)
         LDWA     1,i           ;for (j = 1
         STWA     j,d
for2:    CPWA     numPts,d      ;j <= numPts
         BRGT     endFor2
         @DECI    value,d       ;scanf(”%d”, &value)
         LDWA     value,d       ;move value
         STWA     -2,s
         SUBSP    2,i           ;push #n
         CALL     printBar      ;printBar(value)
         ADDSP    2,i           ;pop #n
         LDWA     j,d           ;j++)
         ADDA     1,i
         STWA     j,d
         BR       for2
endFor2: RET
```

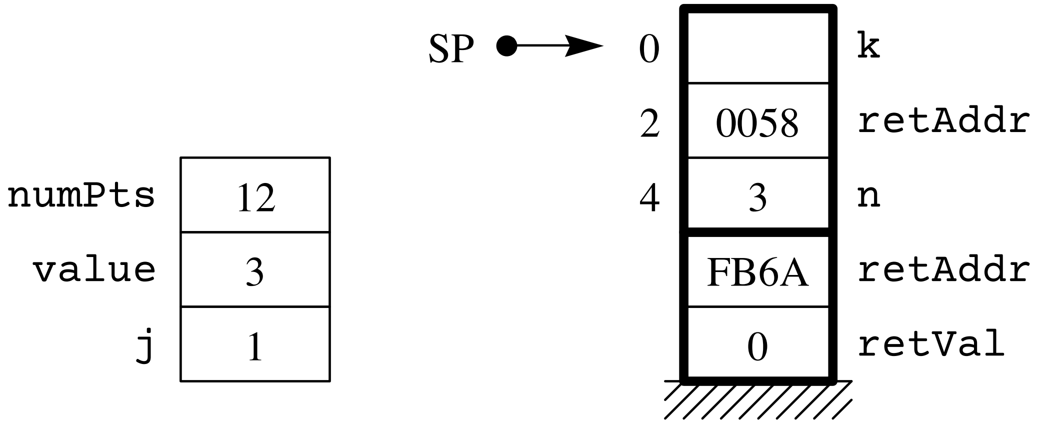
```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2              ;global variable #2d
value:   .BLOCK  2              ;global variable #2d
j:       .BLOCK  2              ;global variable #2d


 ;******* void printBar(int n)
```

Computer Systems Sixth edition Figure 6.29

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
          BR      main
numPts:   .BLOCK  2             ;global variable #2d
value:    .BLOCK  2             ;global variable #2d
j:        .BLOCK  2             ;global variable #2d


;******* void printBar(int n)
```



```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
          BR       main
numPts:   .BLOCK   2             ;global variable #2d
value:    .BLOCK   2             ;global variable #2d
j:        .BLOCK   2             ;global variable #2d


;******* void printBar(int n)
n:        .EQUATE 4              ;formal parameter #2d
```

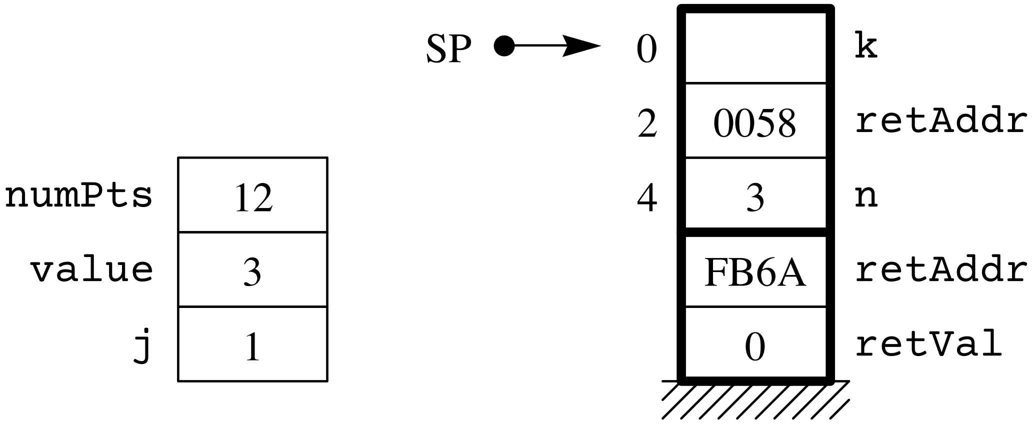
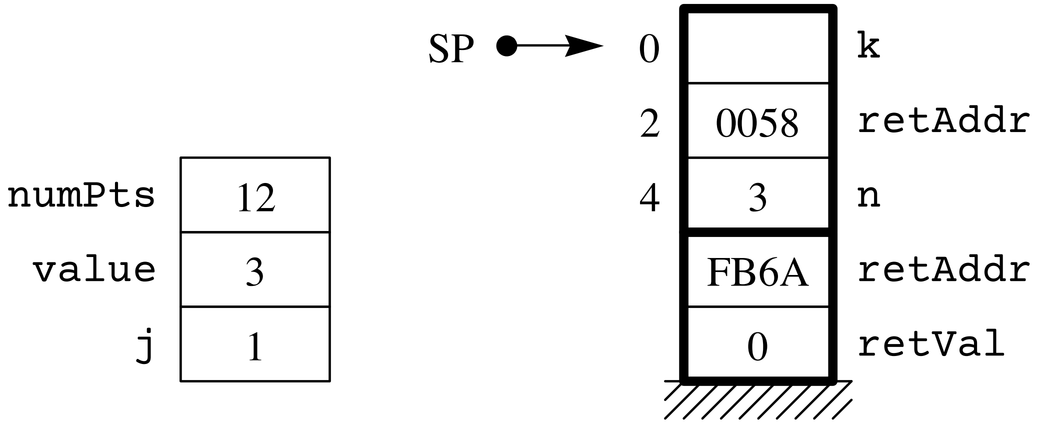


Figure 6.29

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
          BR        main
numPts:   .BLOCK    2              ;global variable #2d
value:    .BLOCK    2              ;global variable #2d
j:        .BLOCK    2              ;global variable #2d


;*******  void printBar(int n)
n:        .EQUATE 4                ;formal parameter #2d
k:        .EQUATE 0                ;local variable #2d
```



```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
          BR      main
numPts:   .BLOCK  2            ;global variable #2d
value:    .BLOCK  2            ;global variable #2d
j:        .BLOCK  2            ;global variable #2d
```

```
;******* void printBar(int n)
n:        .EQUATE 4            ;formal parameter #2d
k:        .EQUATE 0            ;local variable #2d
printBar:SUBSP    2,i          ;push #k
```

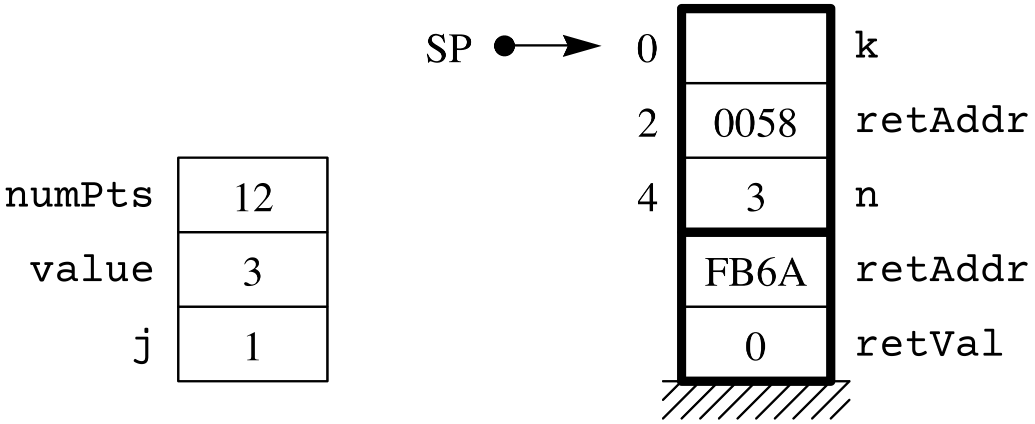


Figure 6.29

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
          BR        main
numPts:   .BLOCK    2              ;global variable #2d
value:    .BLOCK    2              ;global variable #2d
j:        .BLOCK    2              ;global variable #2d


;******* void printBar(int n)
n:        .EQUATE 4                ;formal parameter #2d
k:        .EQUATE 0                ;local variable #2d
printBar:SUBSP    2,i              ;push #k
          LDWA    1,i              ;for (k = 1
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR       main
numPts:  .BLOCK   2              ;global variable #2d
value:   .BLOCK   2              ;global variable #2d
j:       .BLOCK   2              ;global variable #2d


;******* void printBar(int n)
n:       .EQUATE 4               ;formal parameter #2d
k:       .EQUATE 0               ;local variable #2d
printBar:SUBSP    2,i            ;push #k
         LDWA     1,i            ;for (k = 1
         STWA     k,s
```

Figure 6.29

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2             ;global variable #2d
value:   .BLOCK  2             ;global variable #2d
j:       .BLOCK  2             ;global variable #2d


;******* void printBar(int n)
n:       .EQUATE 4             ;formal parameter #2d
k:       .EQUATE 0             ;local variable #2d
printBar:SUBSP   2,i           ;push #k
         LDWA    1,i           ;for (k = 1
         STWA    k,s
for1:    CPWA    n,s           ;k <= n
```

Figure 6.29

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2             ;global variable #2d
value:   .BLOCK  2             ;global variable #2d
j:       .BLOCK  2             ;global variable #2d


;******* void printBar(int n)
n:       .EQUATE 4             ;formal parameter #2d
k:       .EQUATE 0             ;local variable #2d
printBar:SUBSP   2,i           ;push #k
         LDWA    1,i           ;for (k = 1
         STWA    k,s
for1:    CPWA    n,s           ;k <= n
         BRGT    endFor1
```

Figure 6.29

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
          BR       main
numPts:   .BLOCK   2             ;global variable #2d
value:    .BLOCK   2             ;global variable #2d
j:        .BLOCK   2             ;global variable #2d


;******* void printBar(int n)
n:       .EQUATE 4              ;formal parameter #2d
k:       .EQUATE 0              ;local variable #2d
printBar:SUBSP   2,i            ;push #k
         LDWA    1,i            ;for (k = 1
         STWA    k,s
for1:    CPWA    n,s            ;k <= n
         BRGT    endFor1
         @CHARO  ’*’,i          ;printf(”*”)
```

Figure 6.29

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(″*″);
    }
    printf(″\n″);
}

int main() {
    scanf(″%d″, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(″%d″, &value);
        printBar(value);
    }
    return 0;
}
```

```
          BR      main
numPts:   .BLOCK  2             ;global variable #2d
value:    .BLOCK  2             ;global variable #2d
j:        .BLOCK  2             ;global variable #2d


;******* void printBar(int n)
n:        .EQUATE 4             ;formal parameter #2d
k:        .EQUATE 0             ;local variable #2d
printBar:SUBSP    2,i           ;push #k
          LDWA    1,i           ;for (k = 1
          STWA    k,s
for1:     CPWA    n,s           ;k <= n
          BRGT    endFor1
          @CHARO  ′*′,i         ;printf(″*″)
          LDWA    k,s           ;k++)
```

```c
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
          BR       main
numPts:   .BLOCK   2              ;global variable #2d
value:    .BLOCK   2              ;global variable #2d
j:        .BLOCK   2              ;global variable #2d


;******* void printBar(int n)
n:        .EQUATE 4               ;formal parameter #2d
k:        .EQUATE 0               ;local variable #2d
printBar:SUBSP    2,i             ;push #k
          LDWA    1,i             ;for (k = 1
          STWA    k,s
for1:     CPWA    n,s             ;k <= n
          BRGT    endFor1
          @CHARO  ’*’,i           ;printf(”*”)
          LDWA    k,s             ;k++)
          ADDA    1,i
```

Figure 6.29

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2            ;global variable #2d
value:   .BLOCK  2            ;global variable #2d
j:       .BLOCK  2            ;global variable #2d


;******* void printBar(int n)
n:       .EQUATE 4            ;formal parameter #2d
k:       .EQUATE 0            ;local variable #2d
printBar:SUBSP   2,i          ;push #k
         LDWA    1,i          ;for (k = 1
         STWA    k,s
for1:    CPWA    n,s          ;k <= n
         BRGT    endFor1
         @CHARO  ’*’,i        ;printf(”*”)
         LDWA    k,s          ;k++)
         ADDA    1,i
         STWA    k,s
```

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(″*″);
    }
    printf(″\n″);
}

int main() {
    scanf(″%d″, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(″%d″, &value);
        printBar(value);
    }
    return 0;
}
```

```
          BR        main
numPts:   .BLOCK    2               ;global variable #2d
value:    .BLOCK    2               ;global variable #2d
j:        .BLOCK    2               ;global variable #2d


;******* void printBar(int n)
n:        .EQUATE 4                 ;formal parameter #2d
k:        .EQUATE 0                 ;local variable #2d
printBar:SUBSP    2,i               ;push #k
          LDWA    1,i               ;for (k = 1
          STWA    k,s
for1:     CPWA    n,s               ;k <= n
          BRGT    endFor1
          @CHARO  ′*′,i             ;printf(″*″)
          LDWA    k,s               ;k++)
          ADDA    1,i
          STWA    k,s
          BR      for1
```

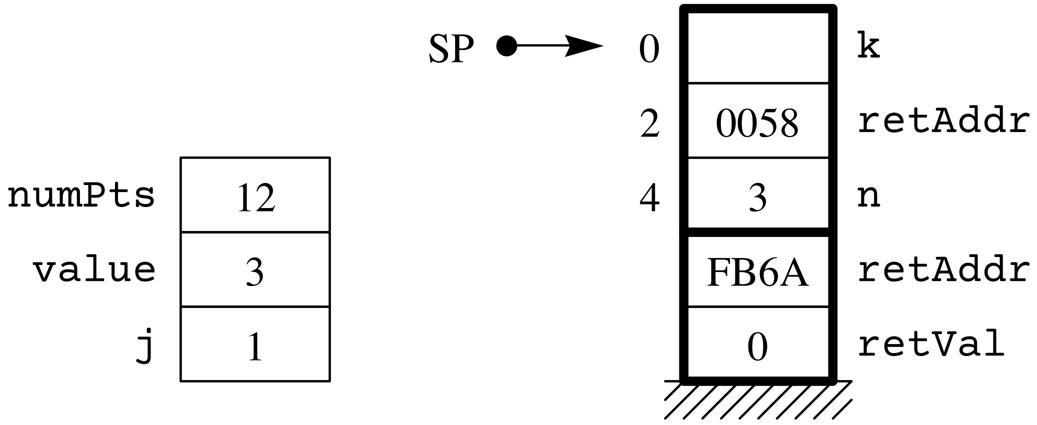
Figure 6.29

```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(″*″);
    }
    printf(″\n″);
}

int main() {
    scanf(″%d″, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(″%d″, &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2             ;global variable #2d
value:   .BLOCK  2             ;global variable #2d
j:       .BLOCK  2             ;global variable #2d



;******* void printBar(int n)
n:       .EQUATE 4             ;formal parameter #2d
k:       .EQUATE 0             ;local variable #2d
printBar:SUBSP   2,i           ;push #k
         LDWA    1,i           ;for (k = 1
         STWA    k,s
for1:    CPWA    n,s           ;k <= n
         BRGT    endFor1
         @CHARO  ′*′,i         ;printf(″*″)
         LDWA    k,s           ;k++)
         ADDA    1,i
         STWA    k,s
         BR      for1
endFor1: @CHARO  ′\n′,i        ;printf(″\n″)
```

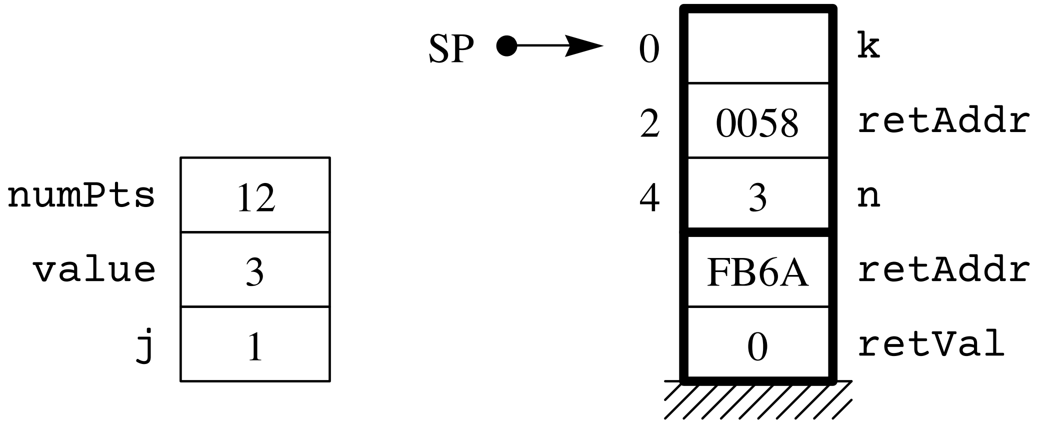
```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2             ;global variable #2d
value:   .BLOCK  2             ;global variable #2d
j:       .BLOCK  2             ;global variable #2d


;******* void printBar(int n)
n:       .EQUATE 4             ;formal parameter #2d
k:       .EQUATE 0             ;local variable #2d
printBar:SUBSP   2,i           ;push #k
         LDWA    1,i           ;for (k = 1
         STWA    k,s
for1:    CPWA    n,s           ;k <= n
         BRGT    endFor1
         @CHARO  '*',i         ;printf("*")
         LDWA    k,s           ;k++)
         ADDA    1,i
         STWA    k,s
         BR      for1
endFor1: @CHARO  '\n',i        ;printf("\n")
```



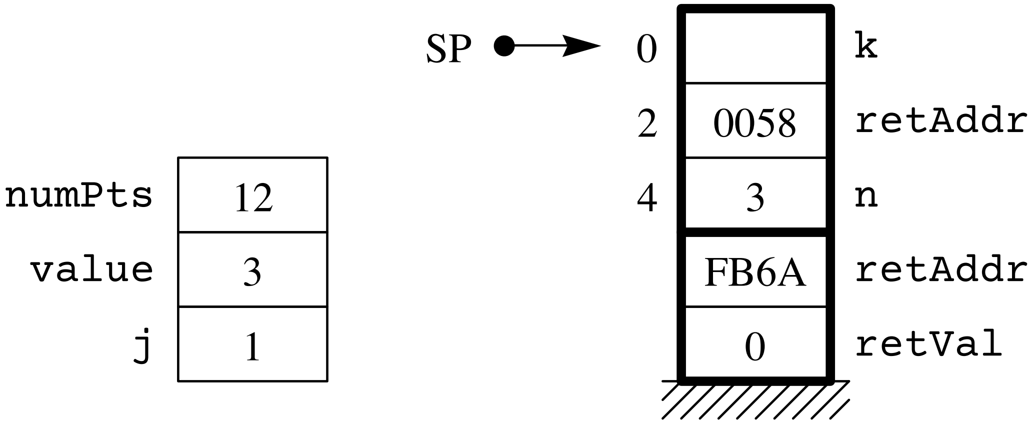
```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR       main
numPts:  .BLOCK   2             ;global variable #2d
value:   .BLOCK   2             ;global variable #2d
j:       .BLOCK   2             ;global variable #2d


;******* void printBar(int n)
n:       .EQUATE  4             ;formal parameter #2d
k:       .EQUATE  0             ;local variable #2d
printBar:SUBSP    2,i           ;push #k
         LDWA     1,i           ;for (k = 1
         STWA     k,s
for1:    CPWA     n,s           ;k <= n
         BRGT     endFor1
         @CHARO   '*',i         ;printf("*")
         LDWA     k,s           ;k++)
         ADDA     1,i
         STWA     k,s
         BR       for1
endFor1: @CHARO   '\n',i        ;printf("\n")
         ADDSP    2,i           ;pop #k
```



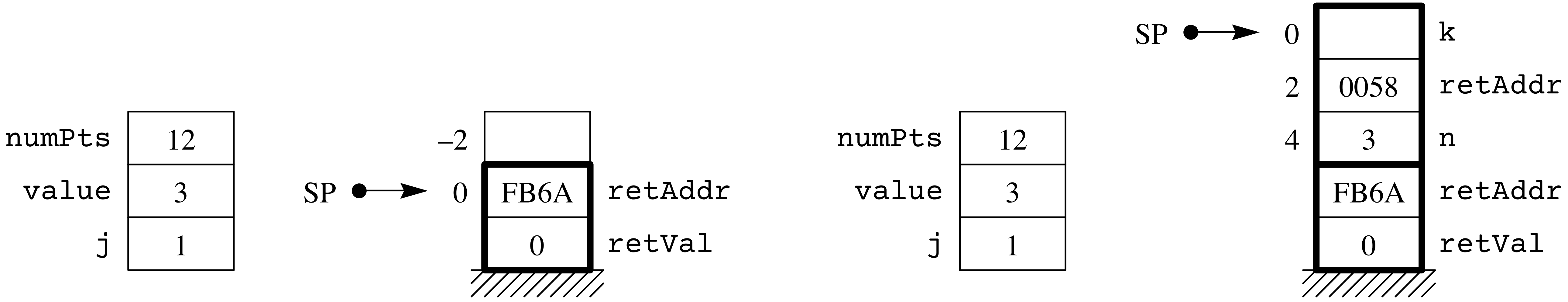
```
#include <stdio.h>

int numPts;
int value;
int j;

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
numPts:  .BLOCK  2             ;global variable #2d
value:   .BLOCK  2             ;global variable #2d
j:       .BLOCK  2             ;global variable #2d


;******* void printBar(int n)
n:       .EQUATE 4             ;formal parameter #2d
k:       .EQUATE 0             ;local variable #2d
printBar:SUBSP   2,i           ;push #k
         LDWA    1,i           ;for (k = 1
         STWA    k,s
for1:    CPWA    n,s           ;k <= n
         BRGT    endFor1
         @CHARO  '*',i         ;printf("*")
         LDWA    k,s           ;k++)
         ADDA    1,i
         STWA    k,s
         BR      for1
endFor1: @CHARO  '\n',i        ;printf("\n")
         ADDSP   2,i           ;pop #k
         RET
```



Computer Systems Sixth edition

Figure 6.30



(a) After `scanf("%d", &value)`.

(b) After `printBar(value)`.

Function call with local variables

- To access the actual parameter in the caller, use stack-relative addressing (s)
- To access the formal parameter in the callee, use stack-relative addressing (s)

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    int numPts;
    int value;
    int j;
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
```

Code for `printBar()`

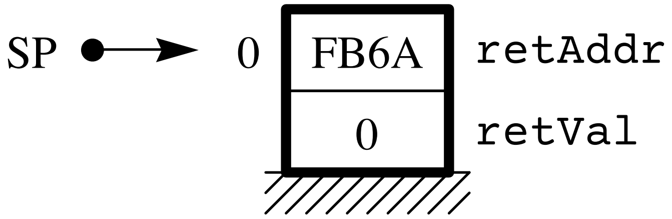
```
;******* main()
```

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR        main
```

Code for `printBar()`

```
;******* main()
```



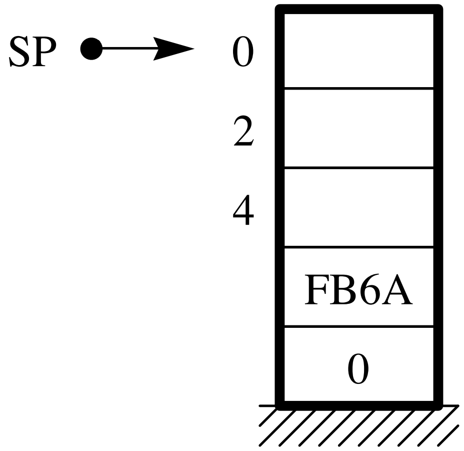
(a) Before `SUBSP 6,i`

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
```

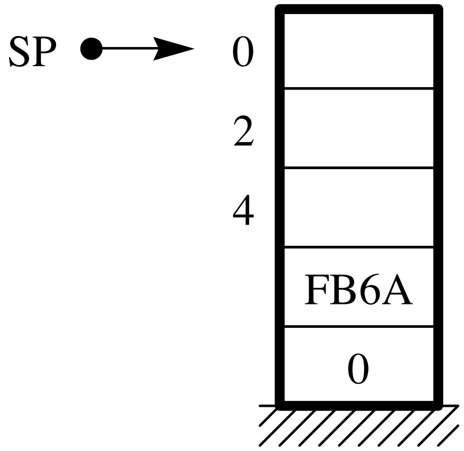
Code for `printBar()`

```
;******* main()
```



(b) After `SUBSP 6,i`

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```



(b) After `SUBSP 6,i`

```
         BR      main
```

Code for `printBar()`

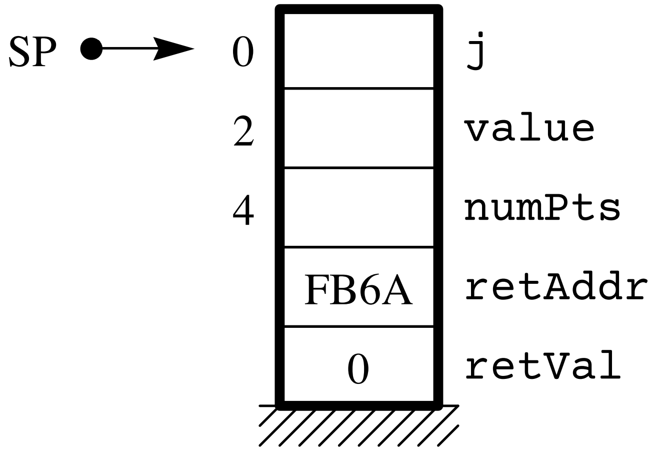
```
;******* main()
numPts:  .EQUATE 4           ;local variable #2d
```

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
```

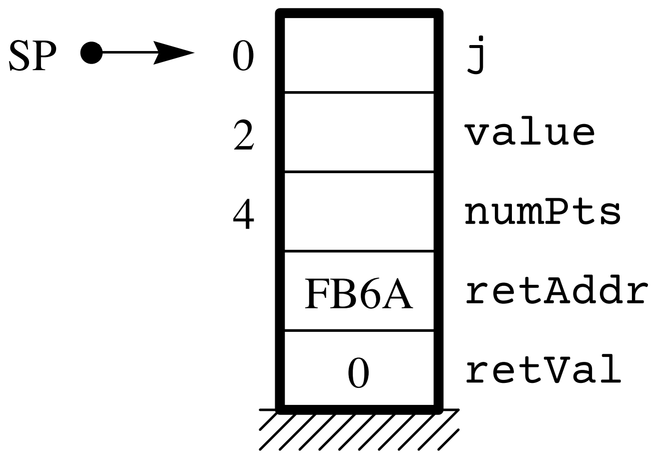
Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4          ;local variable #2d
value:   .EQUATE 2          ;local variable #2d
```



(b) After `SUBSP 6,i`

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```



(b) After `SUBSP 6,i`

```
         BR        main
```

Code for `printBar()`

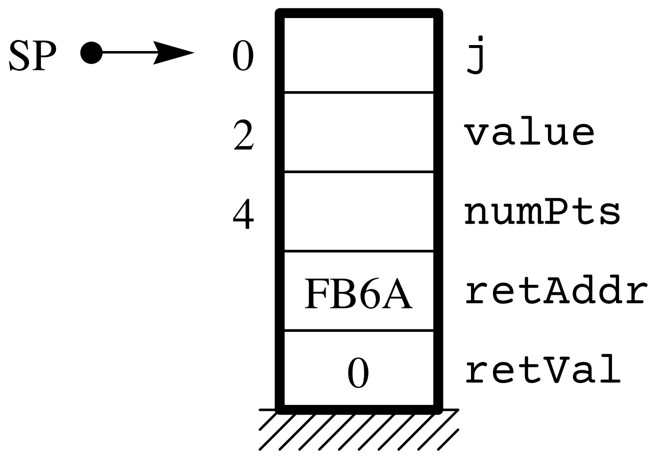
```
;******* main()
numPts:  .EQUATE 4           ;local variable #2d
value:   .EQUATE 2           ;local variable #2d
j:       .EQUATE 0           ;local variable #2d
```

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
```

Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4          ;local variable #2d
value:   .EQUATE 2          ;local variable #2d
j:       .EQUATE 0          ;local variable #2d
main:    SUBSP   6,i        ;push #numPts #value #j
```



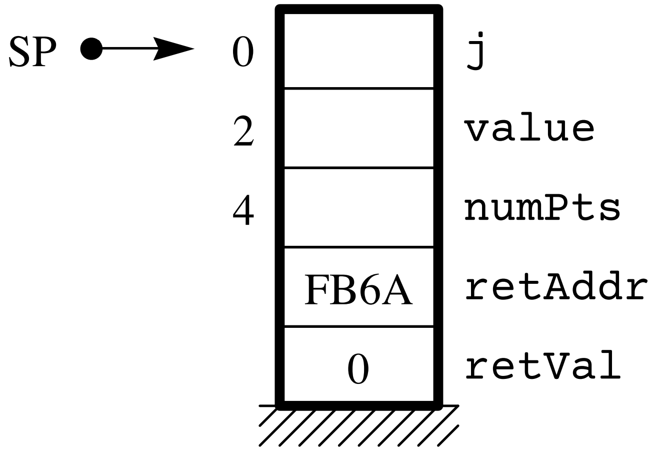
(b) After `SUBSP 6,i`

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
```

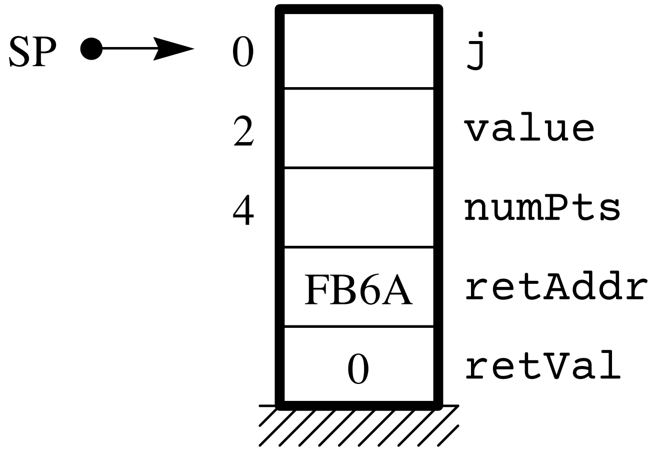
Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4          ;local variable #2d
value:   .EQUATE 2          ;local variable #2d
j:       .EQUATE 0          ;local variable #2d
main:    SUBSP   6,i        ;push #numPts #value #j
         @DECI   numPts,s   ;scanf("%d", &numPts)
```



(b) After `SUBSP 6,i`

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```



(b) After `SUBSP 6,i`

```
         BR      main
```

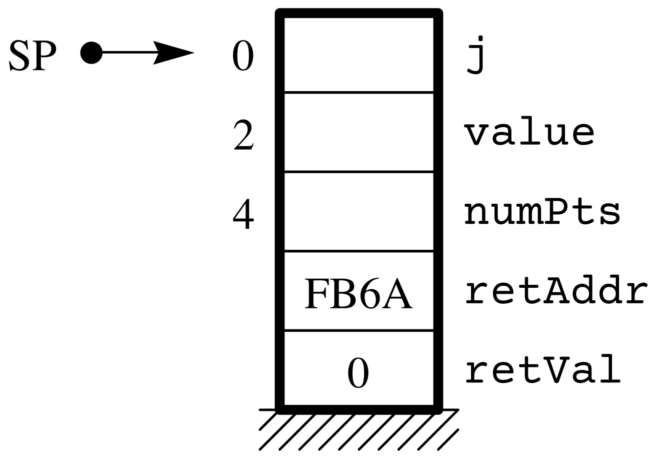
Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4         ;local variable #2d
value:   .EQUATE 2         ;local variable #2d
j:       .EQUATE 0         ;local variable #2d
main:    SUBSP   6,i       ;push #numPts #value #j
         @DECI   numPts,s  ;scanf("%d", &numPts)
         LDWA    1,i       ;for (j = 1
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
```

Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4           ;local variable #2d
value:   .EQUATE 2           ;local variable #2d
j:       .EQUATE 0           ;local variable #2d
main:    SUBSP   6,i         ;push #numPts #value #j
         @DECI   numPts,s    ;scanf("%d", &numPts)
         LDWA    1,i         ;for (j = 1
         STWA    j,s
```



(b) After `SUBSP 6,i`

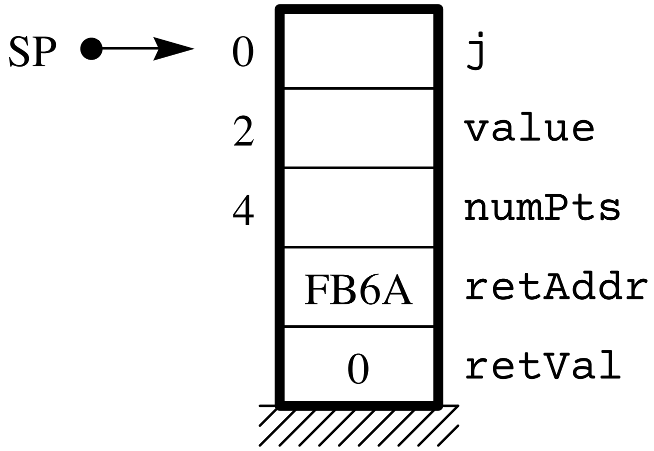
Figure 6.31

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    int numPts;
    int value;
    int j;
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
```

Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4          ;local variable #2d
value:   .EQUATE 2          ;local variable #2d
j:       .EQUATE 0          ;local variable #2d
main:    SUBSP   6,i        ;push #numPts #value #j
         @DECI   numPts,s   ;scanf(”%d”, &numPts)
         LDWA    1,i        ;for (j = 1
         STWA    j,s
for2:    CPWA    numPts,s   ;j <= numPts
```



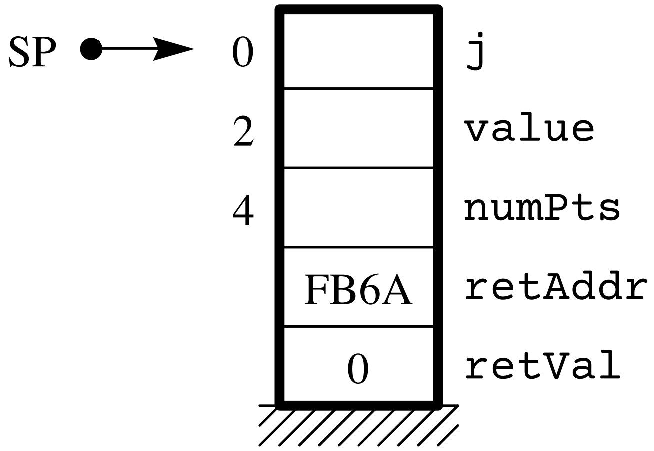
(b) After `SUBSP 6,i`

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
```

Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4           ;local variable #2d
value:   .EQUATE 2           ;local variable #2d
j:       .EQUATE 0           ;local variable #2d
main:    SUBSP   6,i         ;push #numPts #value #j
         @DECI   numPts,s    ;scanf("%d", &numPts)
         LDWA    1,i         ;for (j = 1
         STWA    j,s
for2:    CPWA    numPts,s    ;j <= numPts
         BRGT    endFor2
```



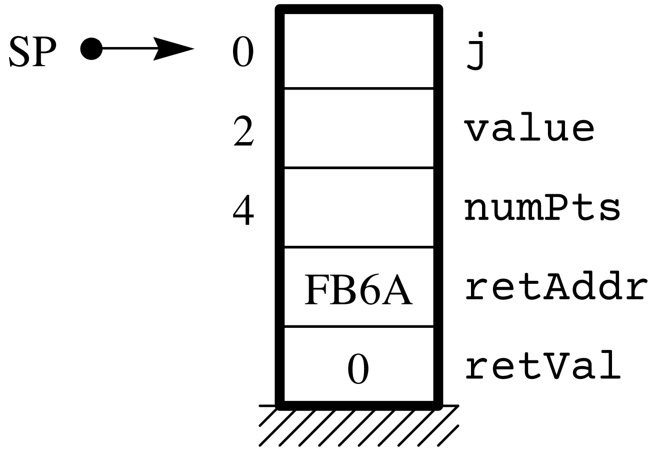
(b) After `SUBSP 6,i`

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

```
         BR      main
```

Code for `printBar()`

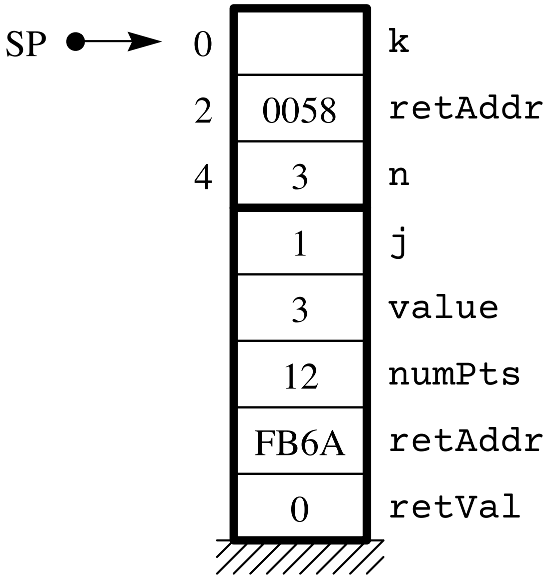
```
;******* main()
numPts:  .EQUATE 4             ;local variable #2d
value:   .EQUATE 2             ;local variable #2d
j:       .EQUATE 0             ;local variable #2d
main:    SUBSP   6,i           ;push #numPts #value #j
         @DECI   numPts,s      ;scanf("%d", &numPts)
         LDWA    1,i           ;for (j = 1
         STWA    j,s
for2:    CPWA    numPts,s      ;j <= numPts
         BRGT    endFor2
         @DECI   value,s       ;scanf("%d", &value)
```



(b) After `SUBSP 6,i`

Computer Systems Sixth edition Figure 6.31

```c
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

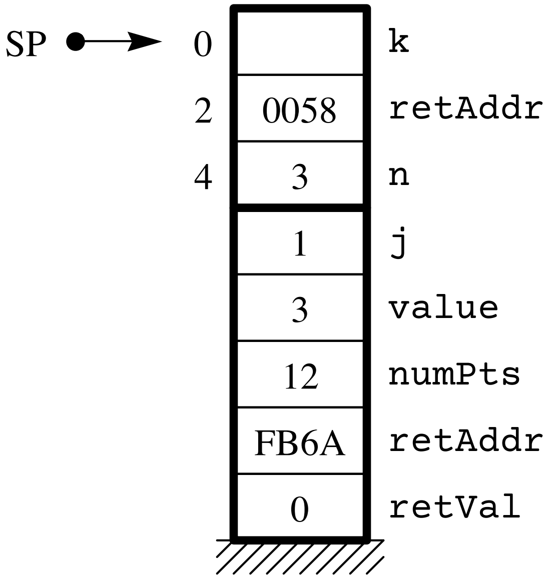


```
         BR      main
```

Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4           ;local variable #2d
value:   .EQUATE 2           ;local variable #2d
j:       .EQUATE 0           ;local variable #2d
main:    SUBSP   6,i         ;push #numPts #value #j
         @DECI   numPts,s    ;scanf("%d", &numPts)
         LDWA    1,i         ;for (j = 1
         STWA    j,s
for2:    CPWA    numPts,s    ;j <= numPts
         BRGT    endFor2
         @DECI   value,s     ;scanf("%d", &value)
```

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

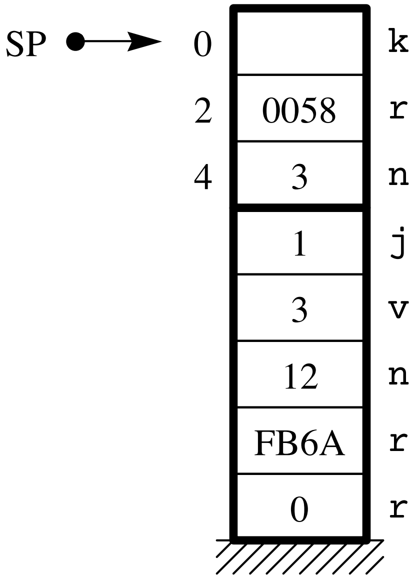


```
         BR      main
```

Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4             ;local variable #2d
value:   .EQUATE 2             ;local variable #2d
j:       .EQUATE 0             ;local variable #2d
main:    SUBSP   6,i           ;push #numPts #value #j
         @DECI   numPts,s      ;scanf("%d", &numPts)
         LDWA    1,i           ;for (j = 1
         STWA    j,s
for2:    CPWA    numPts,s      ;j <= numPts
         BRGT    endFor2
         @DECI   value,s       ;scanf("%d", &value)
         LDWA    value,s       ;move value
```

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf(”*”);
    }
    printf(”\n”);
}

int main() {
    int numPts;
    int value;
    int j;
    scanf(”%d”, &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf(”%d”, &value);
        printBar(value);
    }
    return 0;
}
```



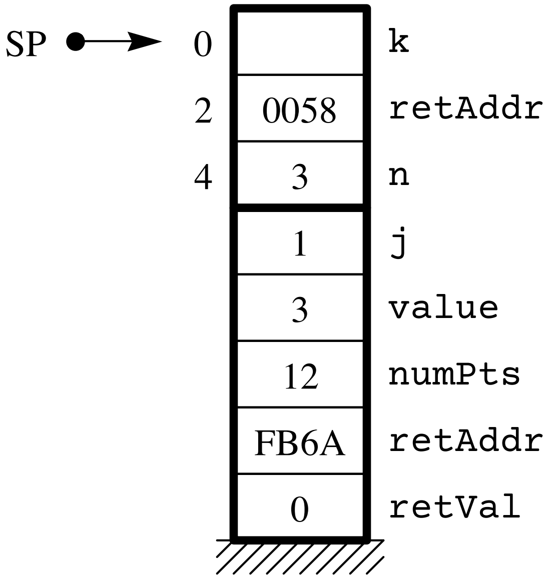
SP → 0		k
2	0058	retAddr
4	3	n
	1	j
	3	value
	12	numPts
	FB6A	retAddr
	0	retVal

```
         BR      main
```

Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4           ;local variable #2d
value:   .EQUATE 2           ;local variable #2d
j:       .EQUATE 0           ;local variable #2d
main:    SUBSP   6,i         ;push #numPts #value #j
         @DECI   numPts,s    ;scanf(”%d”, &numPts)
         LDWA    1,i         ;for (j = 1
         STWA    j,s
for2:    CPWA    numPts,s    ;j <= numPts
         BRGT    endFor2
         @DECI   value,s     ;scanf(”%d”, &value)
         LDWA    value,s     ;move value
         STWA    -2,s
```

```c
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

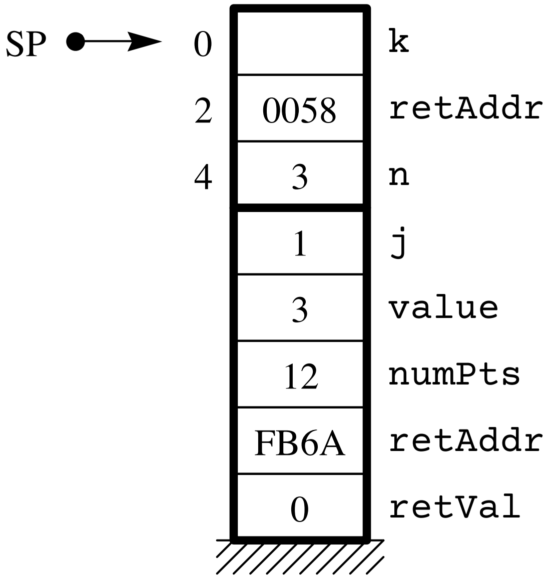


```
         BR      main
```

Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4        ;local variable #2d
value:   .EQUATE 2        ;local variable #2d
j:       .EQUATE 0        ;local variable #2d
main:    SUBSP   6,i      ;push #numPts #value #j
         @DECI   numPts,s ;scanf("%d", &numPts)
         LDWA    1,i      ;for (j = 1
         STWA    j,s
for2:    CPWA    numPts,s ;j <= numPts
         BRGT    endFor2
         @DECI   value,s  ;scanf("%d", &value)
         LDWA    value,s  ;move value
         STWA    -2,s
         SUBSP   2,i      ;push #n
```

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

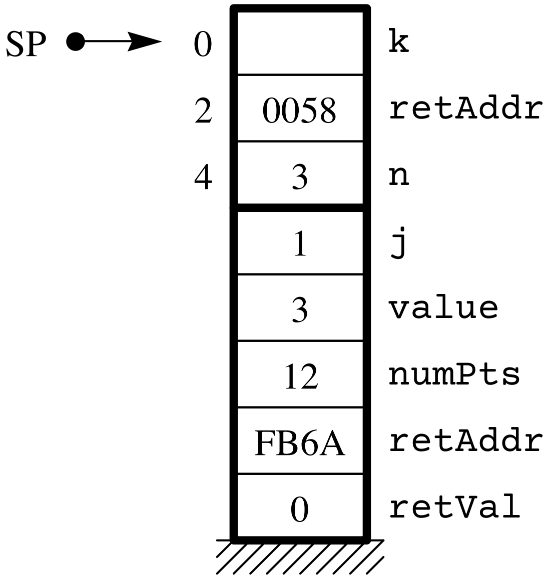


```
         BR      main
```

Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4           ;local variable #2d
value:   .EQUATE 2           ;local variable #2d
j:       .EQUATE 0           ;local variable #2d
main:    SUBSP   6,i         ;push #numPts #value #j
         @DECI   numPts,s    ;scanf("%d", &numPts)
         LDWA    1,i         ;for (j = 1
         STWA    j,s
for2:    CPWA    numPts,s    ;j <= numPts
         BRGT    endFor2
         @DECI   value,s     ;scanf("%d", &value)
         LDWA    value,s     ;move value
         STWA    -2,s
         SUBSP   2,i         ;push #n
         CALL    printBar    ;printBar(value)
```

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

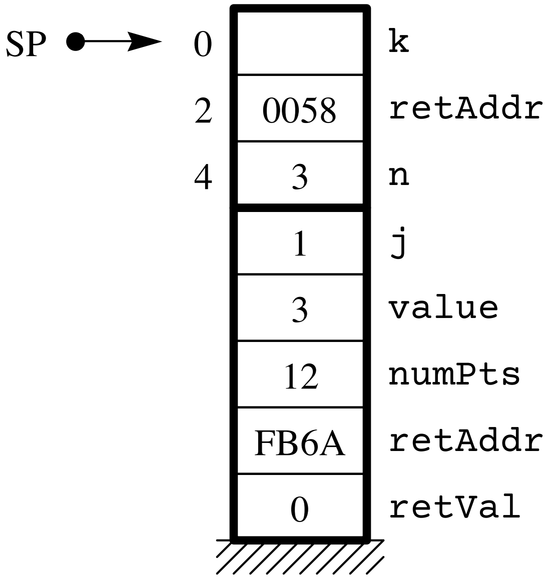


```
         BR      main
```

Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4           ;local variable #2d
value:   .EQUATE 2           ;local variable #2d
j:       .EQUATE 0           ;local variable #2d
main:    SUBSP   6,i         ;push #numPts #value #j
         @DECI   numPts,s    ;scanf("%d", &numPts)
         LDWA    1,i         ;for (j = 1
         STWA    j,s
for2:    CPWA    numPts,s    ;j <= numPts
         BRGT    endFor2
         @DECI   value,s     ;scanf("%d", &value)
         LDWA    value,s     ;move value
         STWA    -2,s
         SUBSP   2,i         ;push #n
         CALL    printBar    ;printBar(value)
         ADDSP   2,i         ;pop #n
```

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

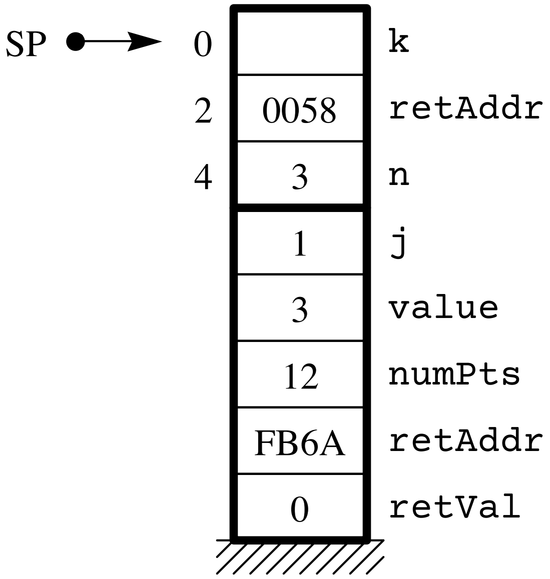


```
         BR      main
```

Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4              ;local variable #2d
value:   .EQUATE 2              ;local variable #2d
j:       .EQUATE 0              ;local variable #2d
main:    SUBSP   6,i            ;push #numPts #value #j
         @DECI   numPts,s       ;scanf("%d", &numPts)
         LDWA    1,i            ;for (j = 1
         STWA    j,s
for2:    CPWA    numPts,s       ;j <= numPts
         BRGT    endFor2
         @DECI   value,s        ;scanf("%d", &value)
         LDWA    value,s        ;move value
         STWA    -2,s
         SUBSP   2,i            ;push #n
         CALL    printBar       ;printBar(value)
         ADDSP   2,i            ;pop #n
         LDWA    j,s            ;j++)
```

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```

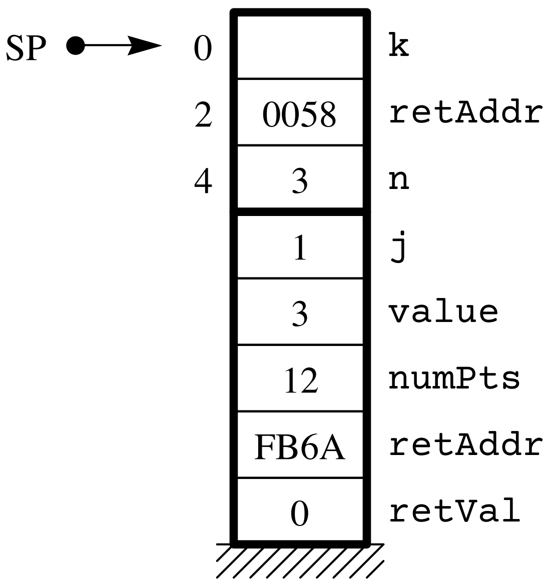


```
         BR      main
```

Code for printBar()

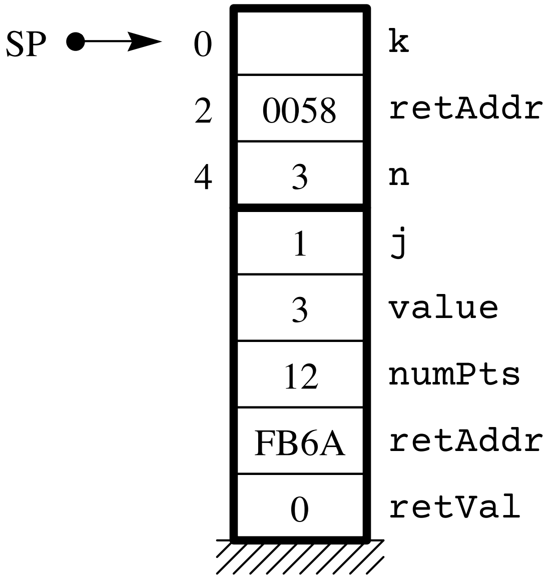
```
;******* main()
numPts:  .EQUATE 4            ;local variable #2d
value:   .EQUATE 2            ;local variable #2d
j:       .EQUATE 0            ;local variable #2d
main:    SUBSP   6,i          ;push #numPts #value #j
         @DECI   numPts,s     ;scanf("%d", &numPts)
         LDWA    1,i          ;for (j = 1
         STWA    j,s
for2:    CPWA    numPts,s     ;j <= numPts
         BRGT    endFor2
         @DECI   value,s      ;scanf("%d", &value)
         LDWA    value,s      ;move value
         STWA    -2,s
         SUBSP   2,i          ;push #n
         CALL    printBar     ;printBar(value)
         ADDSP   2,i          ;pop #n
         LDWA    j,s          ;j++)
         ADDA    1,i
```

```c
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```



```
         BR      main

Code for printBar()

;******* main()
numPts:  .EQUATE 4           ;local variable #2d
value:   .EQUATE 2           ;local variable #2d
j:       .EQUATE 0           ;local variable #2d
main:    SUBSP   6,i         ;push #numPts #value #j
         @DECI   numPts,s    ;scanf("%d", &numPts)
         LDWA    1,i         ;for (j = 1
         STWA    j,s
for2:    CPWA    numPts,s    ;j <= numPts
         BRGT    endFor2
         @DECI   value,s     ;scanf("%d", &value)
         LDWA    value,s     ;move value
         STWA    -2,s
         SUBSP   2,i         ;push #n
         CALL    printBar    ;printBar(value)
         ADDSP   2,i         ;pop #n
         LDWA    j,s         ;j++)
         ADDA    1,i
         STWA    j,s
```

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```



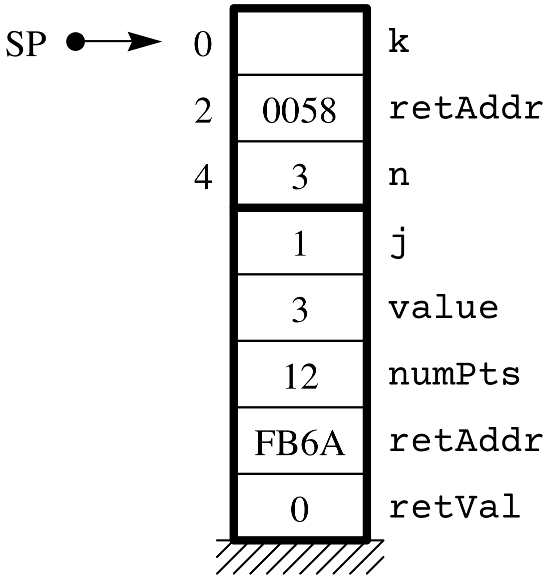
SP →		
0		k
2	0058	retAddr
4	3	n
	1	j
	3	value
	12	numPts
	FB6A	retAddr
	0	retVal

```
         BR      main
```

Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4           ;local variable #2d
value:   .EQUATE 2           ;local variable #2d
j:       .EQUATE 0           ;local variable #2d
main:    SUBSP   6,i         ;push #numPts #value #j
         @DECI   numPts,s    ;scanf("%d", &numPts)
         LDWA    1,i         ;for (j = 1
         STWA    j,s
for2:    CPWA    numPts,s    ;j <= numPts
         BRGT    endFor2
         @DECI   value,s     ;scanf("%d", &value)
         LDWA    value,s     ;move value
         STWA    -2,s
         SUBSP   2,i         ;push #n
         CALL    printBar    ;printBar(value)
         ADDSP   2,i         ;pop #n
         LDWA    j,s         ;j++)
         ADDA    1,i
         STWA    j,s
         BR      for2
```

```
#include <stdio.h>

void printBar(int n) {
    int k;
    for (k = 1; k <= n; k++) {
        printf("*");
    }
    printf("\n");
}

int main() {
    int numPts;
    int value;
    int j;
    scanf("%d", &numPts);
    for (j = 1; j <= numPts; j++) {
        scanf("%d", &value);
        printBar(value);
    }
    return 0;
}
```



SP → 0 | k
2 | 0058 | retAddr
4 | 3 | n
1 | j
3 | value
12 | numPts
FB6A | retAddr
0 | retVal

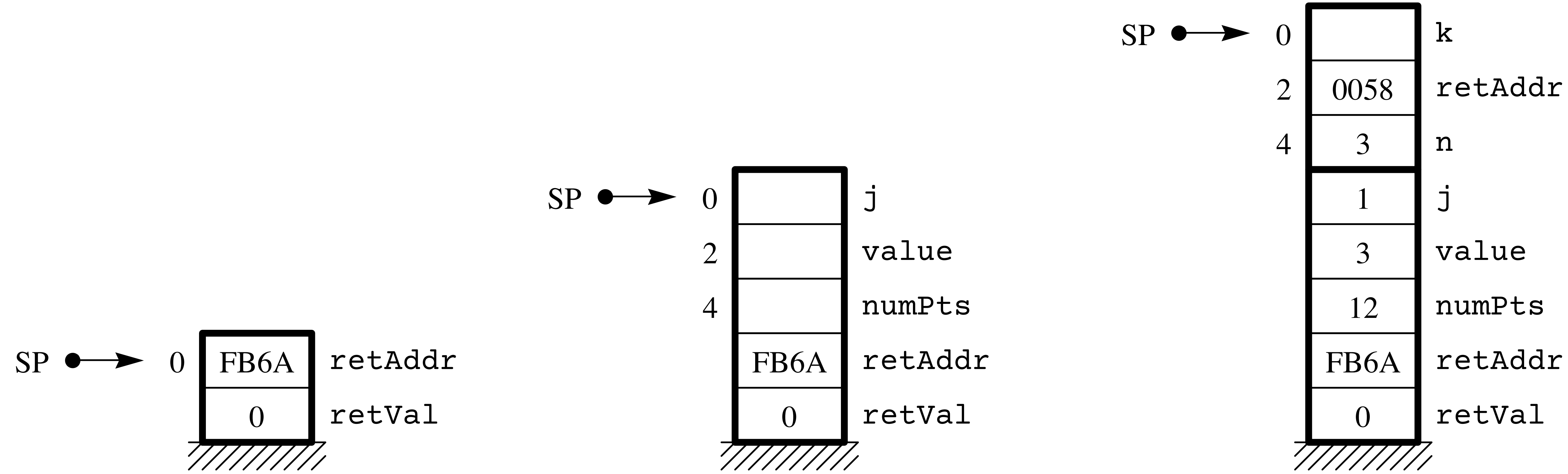
```
         BR      main
```

Code for `printBar()`

```
;******* main()
numPts:  .EQUATE 4             ;local variable #2d
value:   .EQUATE 2             ;local variable #2d
j:       .EQUATE 0             ;local variable #2d
main:    SUBSP   6,i           ;push #numPts #value #j
         @DECI   numPts,s      ;scanf("%d", &numPts)
         LDWA    1,i           ;for (j = 1
         STWA    j,s
for2:    CPWA    numPts,s      ;j <= numPts
         BRGT    endFor2
         @DECI   value,s       ;scanf("%d", &value)
         LDWA    value,s       ;move value
         STWA    -2,s
         SUBSP   2,i           ;push #n
         CALL    printBar      ;printBar(value)
         ADDSP   2,i           ;pop #n
         LDWA    j,s           ;j++)
         ADDA    1,i
         STWA    j,s
         BR      for2
endFor2: ADDSP   6,i           ;pop #j #value #numPts
```

Computer Systems Sixth edition

Figure 6.32



(a) Before `SUBSP 6,i`

(b) After `SUBSP 6,i`

(c) After `printBar(value)`

To call a non-`void` function in C

- Push storage for the return value
- Push the actual parameters
- Push the return address
- Push storage for the local variables

Figure 6.33

```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf(”binCoeff(3, 1) = %d\n”,
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf(”binCoeff(3, 1) = %d\n”,
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
BR        main
```

Code for `binCoeff()`

Figure 6.33

```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf(”binCoeff(3, 1) = %d\n”,
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR       main
```

Code for `binCoeff()`

```
;******* main()
```

```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
```

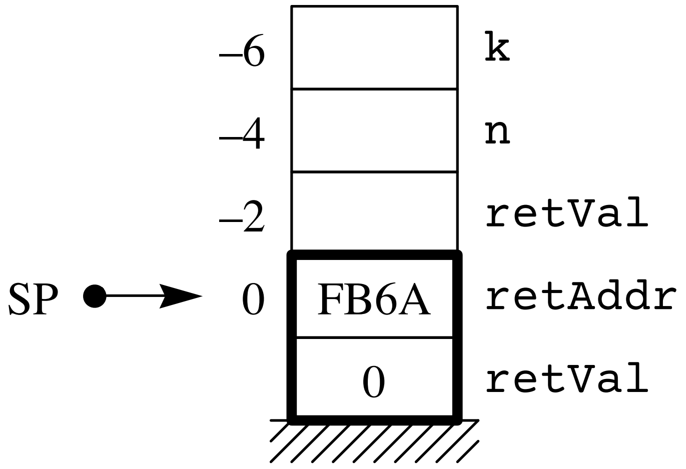
Code for binCoeff()

```
;******* main()
main:    @STRO   msg,d      ;printf("binCoeff(3, 1) = %d\n",
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf(”binCoeff(3, 1) = %d\n”,
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
```

Code for `binCoeff()`

```
;******* main()
main:    @STRO   msg,d        ;printf(”binCoeff(3, 1) = %d\n”,
```

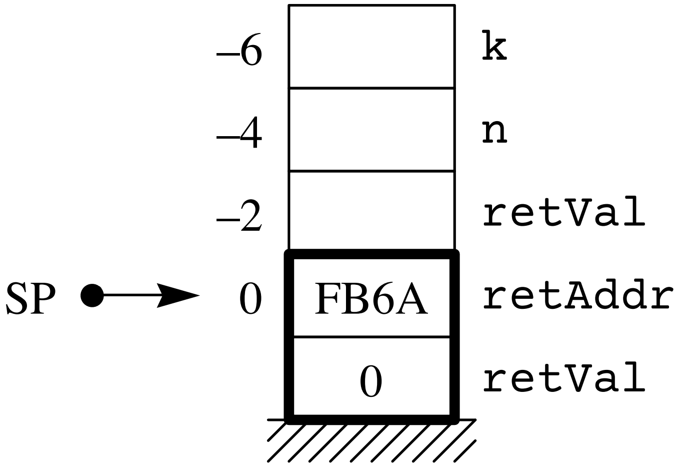


```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
```

Code for `binCoeff()`

```
;******* main()
main:    @STRO   msg,d        ;printf("binCoeff(3, 1) = %d\n",
         LDWA    3,i          ;move 3
```

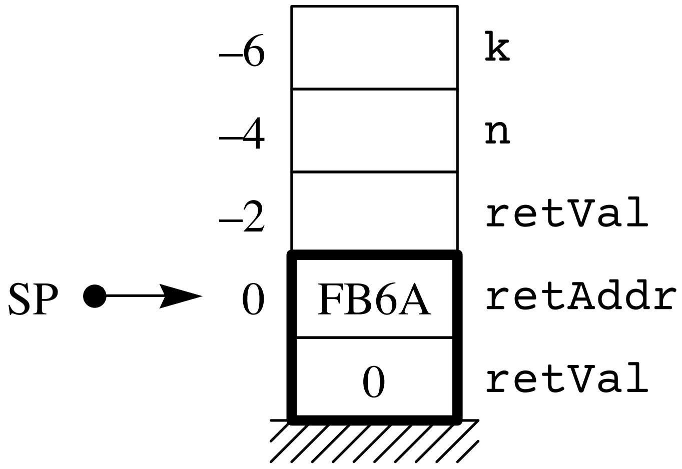


```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
```

Code for `binCoeff()`

```
;******* main()
main:    @STRO   msg,d        ;printf("binCoeff(3, 1) = %d\n",
         LDWA    3,i          ;move 3
         STWA    -4,s
```

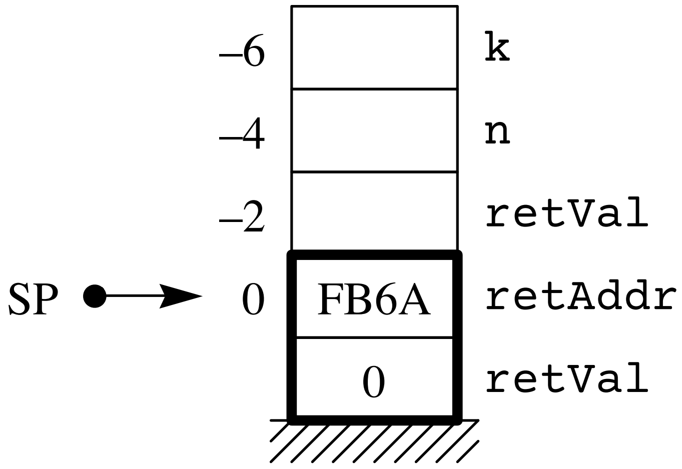


```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf(”binCoeff(3, 1) = %d\n”,
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
```

Code for `binCoeff()`

```
;******* main()
main:    @STRO   msg,d      ;printf(”binCoeff(3, 1) = %d\n”,
         LDWA    3,i        ;move 3
         STWA    -4,s
         LDWA    1,i        ;move 1
```

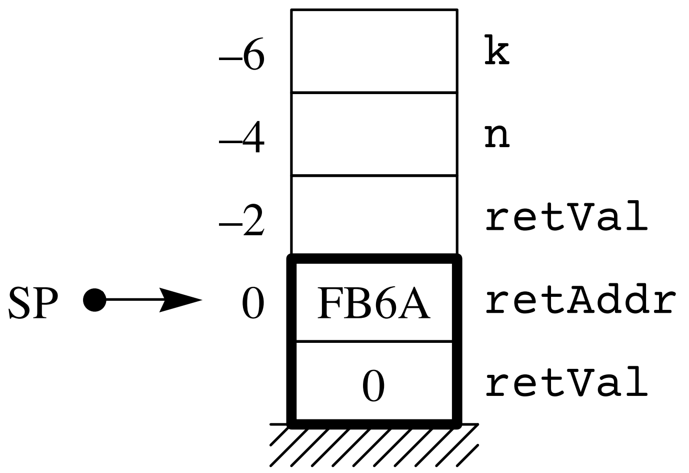


```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf(”binCoeff(3, 1) = %d\n”,
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
```

Code for `binCoeff()`

```
;******* main()
main:    @STRO   msg,d        ;printf(”binCoeff(3, 1) = %d\n”,
         LDWA    3,i          ;move 3
         STWA    -4,s
         LDWA    1,i          ;move 1
         STWA    -6,s
```

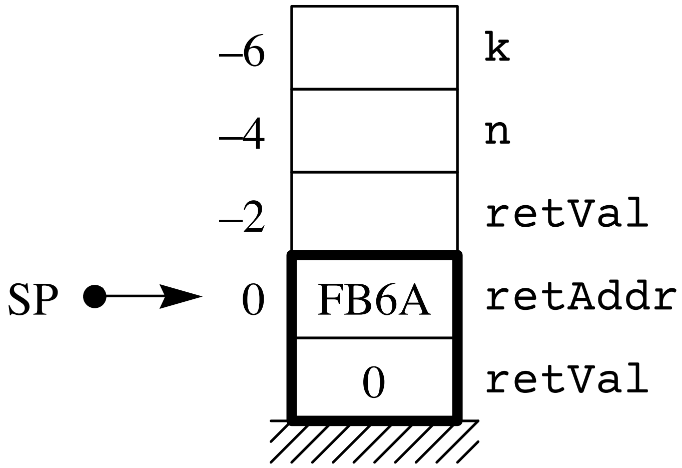


```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
BR      main
```

Code for `binCoeff()`

```
;******* main()
main:    @STRO    msg,d       ;printf("binCoeff(3, 1) = %d\n",
         LDWA     3,i         ;move 3
         STWA     -4,s
         LDWA     1,i         ;move 1
         STWA     -6,s
         SUBSP    6,i         ;push #retVal #n #k
```

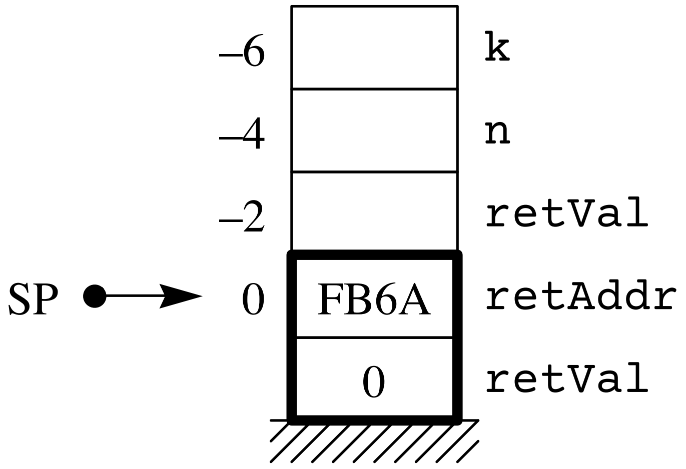


```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
```

Code for `binCoeff()`

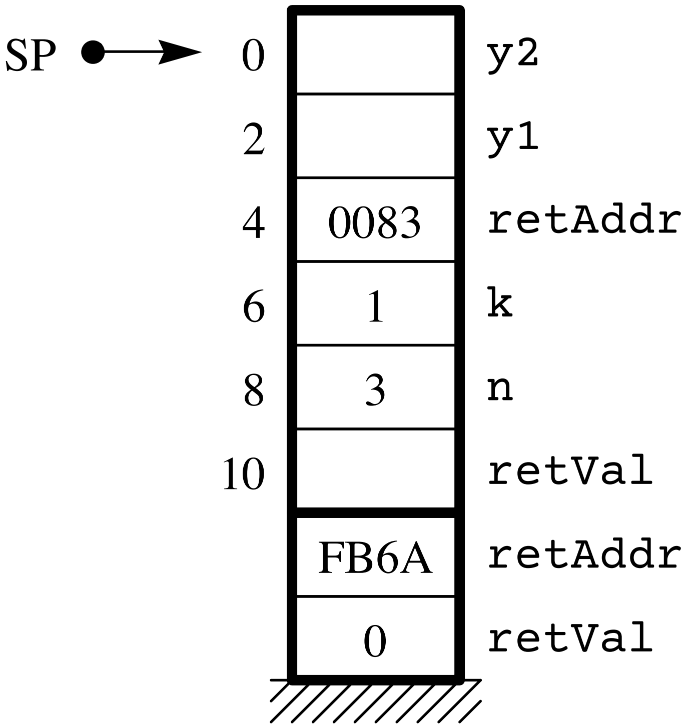
```
;******* main()
main:    @STRO   msg,d        ;printf("binCoeff(3, 1) = %d\n",
         LDWA    3,i          ;move 3
         STWA    -4,s
         LDWA    1,i          ;move 1
         STWA    -6,s
         SUBSP   6,i          ;push #retVal #n #k
```



```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
BR        main
```

Code for `binCoeff()`

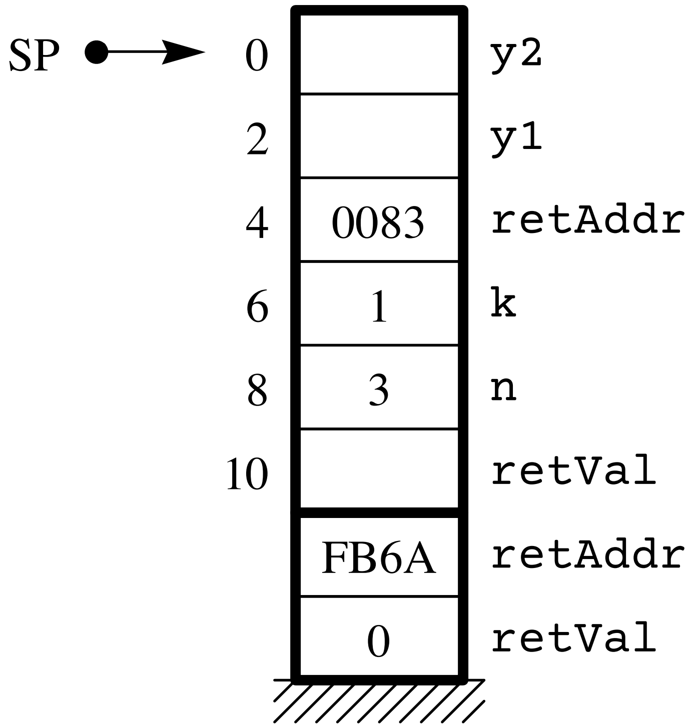


```
;******* main()
main:    @STRO    msg,d        ;printf("binCoeff(3, 1) = %d\n",
         LDWA     3,i          ;move 3
         STWA     -4,s
         LDWA     1,i          ;move 1
         STWA     -6,s
         SUBSP    6,i          ;push #retVal #n #k
         CALL     binCoeff     ;binCoeff(3, 1)
```

```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
BR        main
```

Code for `binCoeff()`

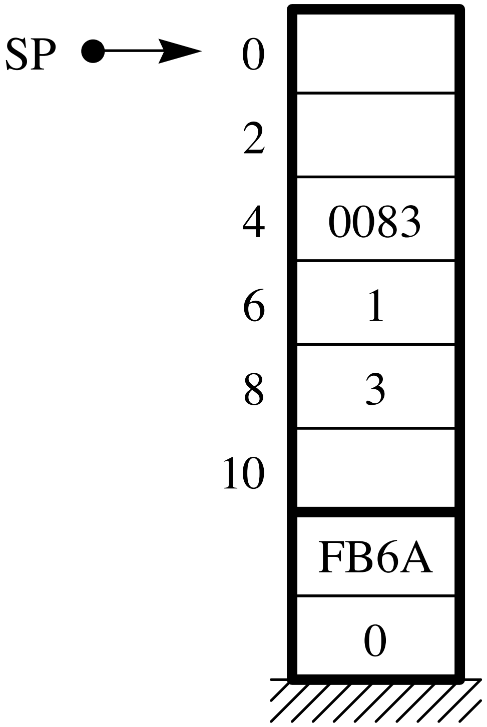


```
;******* main()
main:    @STRO    msg,d       ;printf("binCoeff(3, 1) = %d\n",
         LDWA     3,i         ;move 3
         STWA     -4,s
         LDWA     1,i         ;move 1
         STWA     -6,s
         SUBSP    6,i         ;push #retVal #n #k
         CALL     binCoeff    ;binCoeff(3, 1)
ra1:     ADDSP    6,i         ;pop #k #n #retVal
```

```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf(”binCoeff(3, 1) = %d\n”,
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
BR        main
```

Code for `binCoeff()`

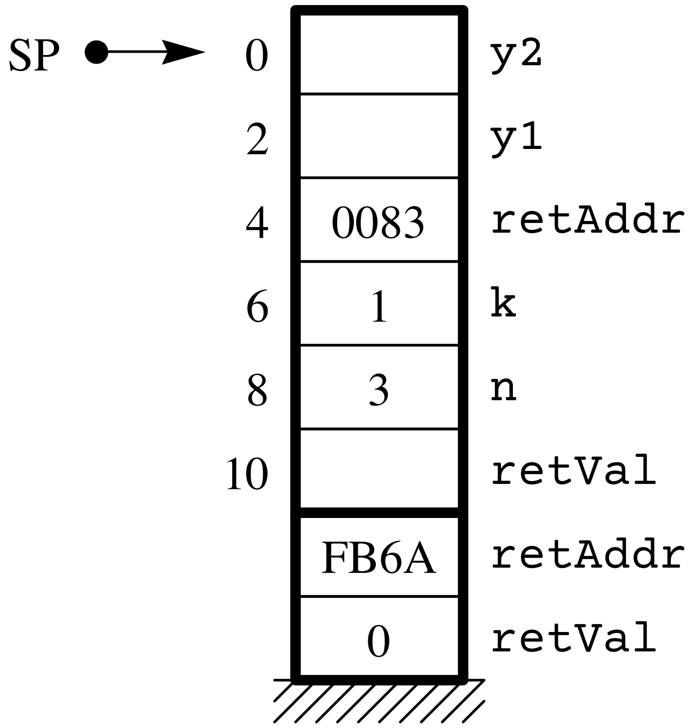


```
;******* main()
main:    @STRO    msg,d       ;printf(”binCoeff(3, 1) = %d\n”,
         LDWA     3,i         ;move 3
         STWA     -4,s
         LDWA     1,i         ;move 1
         STWA     -6,s
         SUBSP    6,i         ;push #retVal #n #k
         CALL     binCoeff    ;binCoeff(3, 1)
ra1:     ADDSP    6,i         ;pop #k #n #retVal
         @DECO    -2,s
```

```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
BR         main
```

Code for binCoeff()

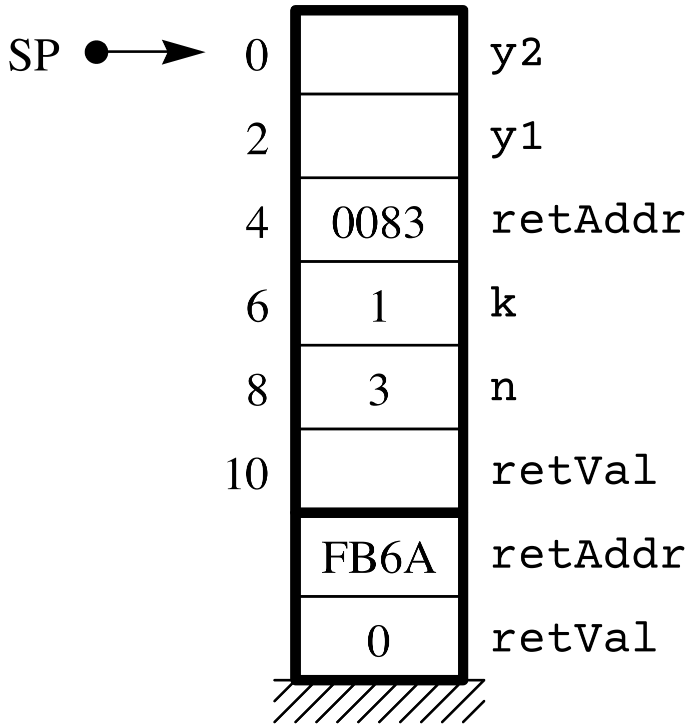


```
;******* main()
main:    @STRO   msg,d       ;printf("binCoeff(3, 1) = %d\n",
         LDWA    3,i         ;move 3
         STWA    -4,s
         LDWA    1,i         ;move 1
         STWA    -6,s
         SUBSP   6,i         ;push #retVal #n #k
         CALL    binCoeff    ;binCoeff(3, 1)
ra1:     ADDSP   6,i         ;pop #k #n #retVal
         @DECO   -2,s
         @CHARO  '\n',i
```

```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf(”binCoeff(3, 1) = %d\n”,
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
BR        main
```

Code for `binCoeff()`



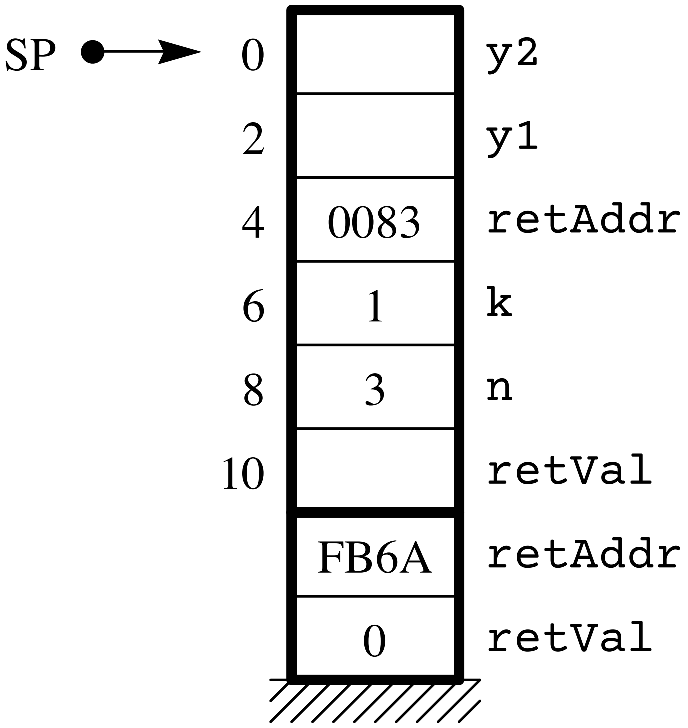
SP →	0		y2
	2		y1
	4	0083	retAddr
	6	1	k
	8	3	n
	10		retVal
		FB6A	retAddr
		0	retVal

```
;******* main()
main:    @STRO    msg,d        ;printf(”binCoeff(3, 1) = %d\n”,
         LDWA     3,i          ;move 3
         STWA     -4,s
         LDWA     1,i          ;move 1
         STWA     -6,s
         SUBSP    6,i          ;push #retVal #n #k
         CALL     binCoeff     ;binCoeff(3, 1)
ra1:     ADDSP    6,i          ;pop #k #n #retVal
         @DECO    -2,s
         @CHARO   ’\n’,i
         RET
```

```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf(”binCoeff(3, 1) = %d\n”,
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
```

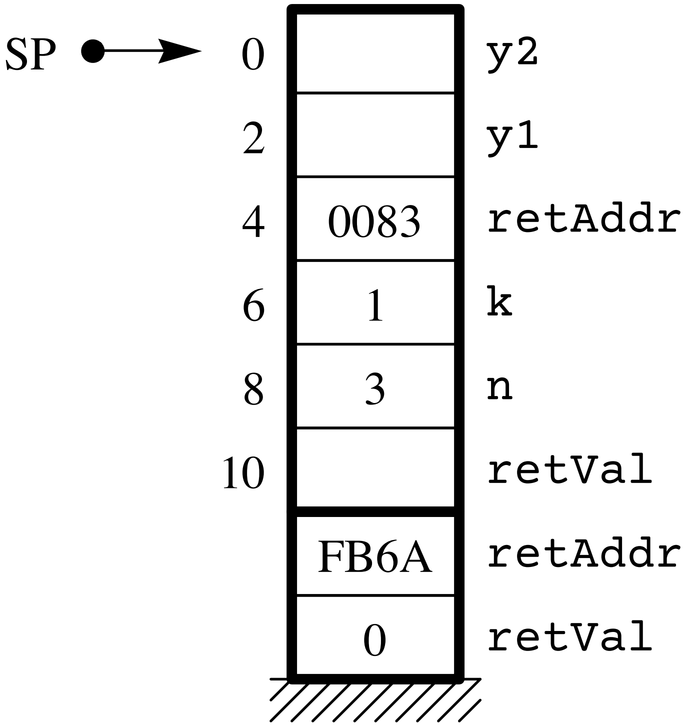
Code for binCoeff()



```
;******* main()
main:    @STRO    msg,d        ;printf(”binCoeff(3, 1) = %d\n”,
         LDWA     3,i          ;move 3
         STWA     -4,s
         LDWA     1,i          ;move 1
         STWA     -6,s
         SUBSP    6,i          ;push #retVal #n #k
         CALL     binCoeff     ;binCoeff(3, 1)
ra1:     ADDSP    6,i          ;pop #k #n #retVal
         @DECO    -2,s
         @CHARO   ’\n’,i
         RET
msg:     .ASCII   ”binCoeff(3, 1) = \0”
```

```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf(”binCoeff(3, 1) = %d\n”,
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
```

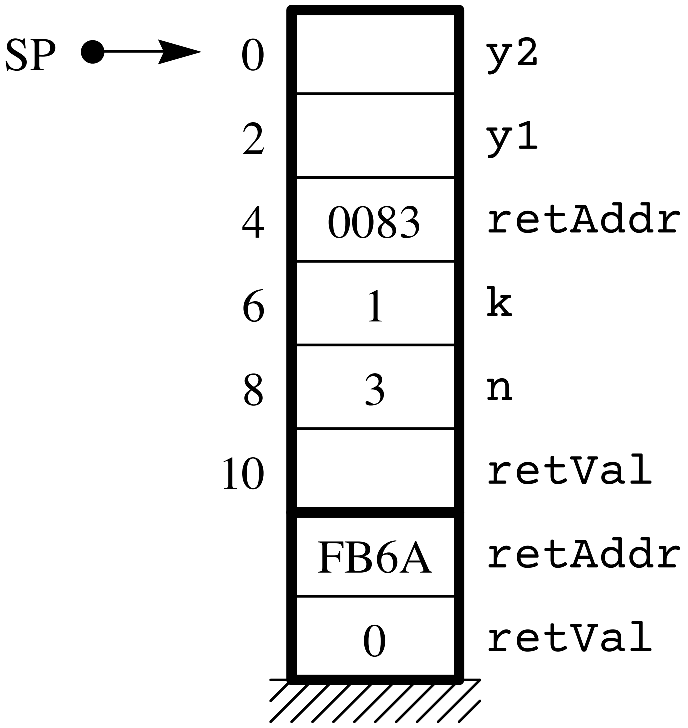


Computer Systems Sixth edition

Figure 6.33

```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10          ;return value #2d
```



```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf(”binCoeff(3, 1) = %d\n”,
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
          BR      main
;
;******* int binCoeff(int n, int k)
retVal:   .EQUATE 10          ;return value #2d
n:        .EQUATE 8           ;formal parameter #2d
```

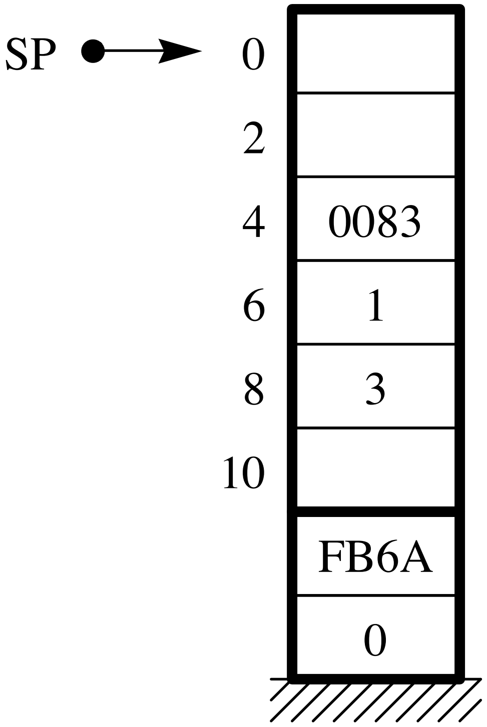
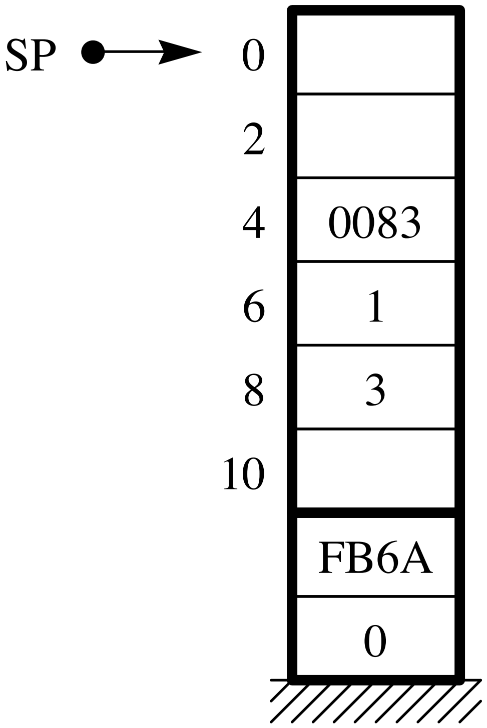


Figure 6.33

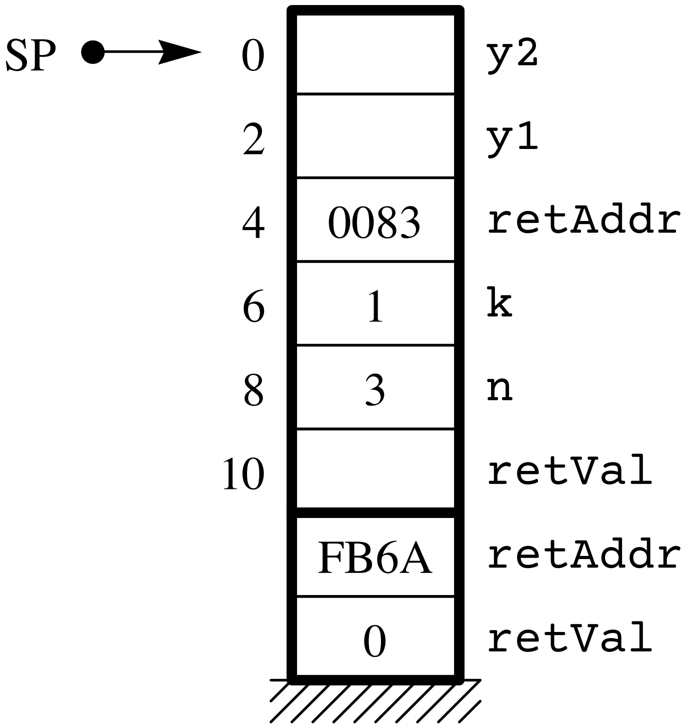
```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10         ;return value #2d
n:       .EQUATE 8          ;formal parameter #2d
k:       .EQUATE 6          ;formal parameter #2d
```



```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
          BR        main
;
;******* int binCoeff(int n, int k)
retVal:   .EQUATE 10          ;return value #2d
n:        .EQUATE 8           ;formal parameter #2d
k:        .EQUATE 6           ;formal parameter #2d
y1:       .EQUATE 2           ;local variable #2d
```



SP →		
0		y2
2		y1
4	0083	retAddr
6	1	k
8	3	n
10		retVal
	FB6A	retAddr
	0	retVal

Figure 6.33

```c
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10          ;return value #2d
n:       .EQUATE 8           ;formal parameter #2d
k:       .EQUATE 6           ;formal parameter #2d
y1:      .EQUATE 2           ;local variable #2d
y2:      .EQUATE 0           ;local variable #2d
```

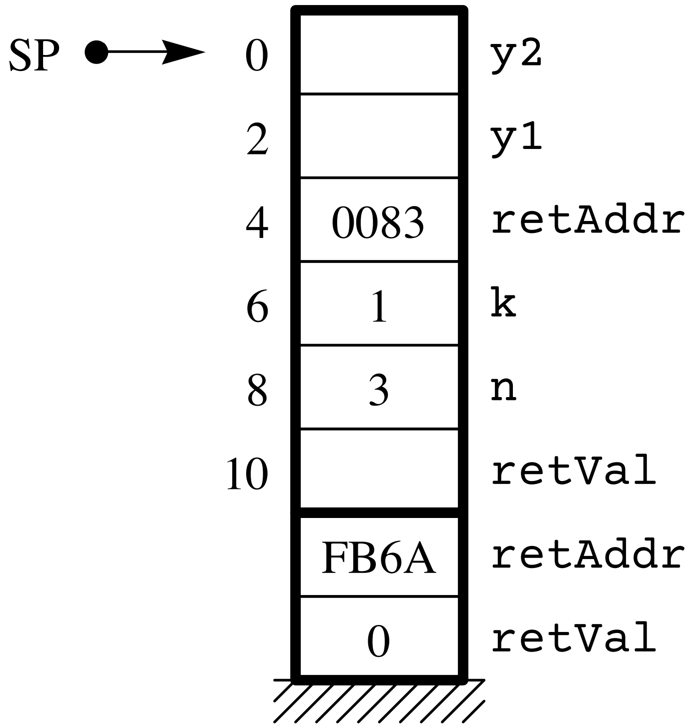
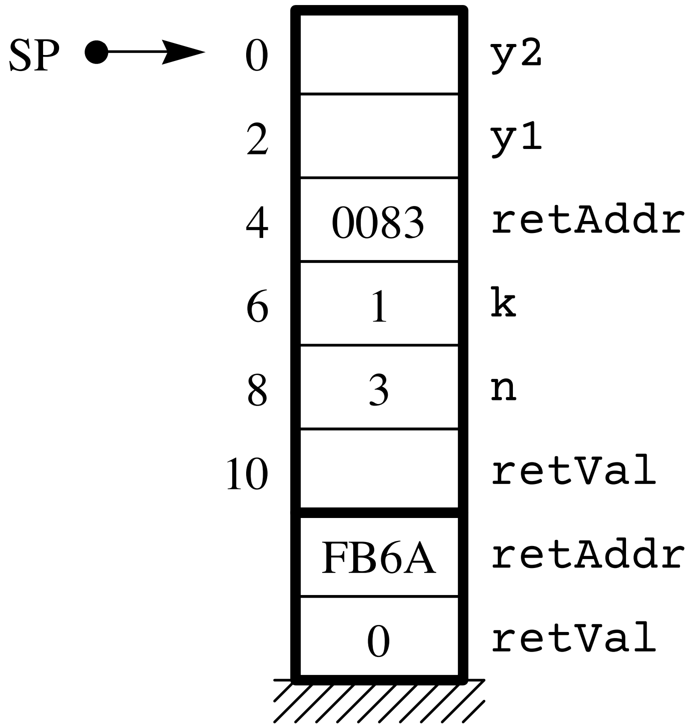


Figure 6.33

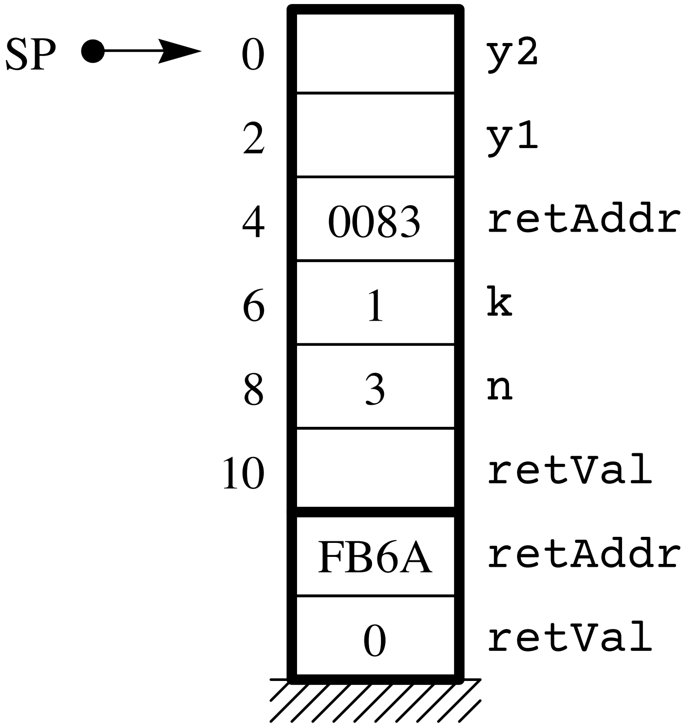
```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf(”binCoeff(3, 1) = %d\n”,
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10          ;return value #2d
n:       .EQUATE 8           ;formal parameter #2d
k:       .EQUATE 6           ;formal parameter #2d
y1:      .EQUATE 2           ;local variable #2d
y2:      .EQUATE 0           ;local variable #2d
binCoeff:SUBSP   4,i         ;push #y1 #y2
```



```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10            ;return value #2d
n:       .EQUATE 8             ;formal parameter #2d
k:       .EQUATE 6             ;formal parameter #2d
y1:      .EQUATE 2             ;local variable #2d
y2:      .EQUATE 0             ;local variable #2d
binCoeff:SUBSP   4,i           ;push #y1 #y2
if:      LDWA    k,s           ;if ((k == 0)
```



```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10            ;return value #2d
n:       .EQUATE 8             ;formal parameter #2d
k:       .EQUATE 6             ;formal parameter #2d
y1:      .EQUATE 2             ;local variable #2d
y2:      .EQUATE 0             ;local variable #2d
binCoeff:SUBSP   4,i           ;push #y1 #y2
if:      LDWA    k,s           ;if ((k == 0)
         BREQ    then
```

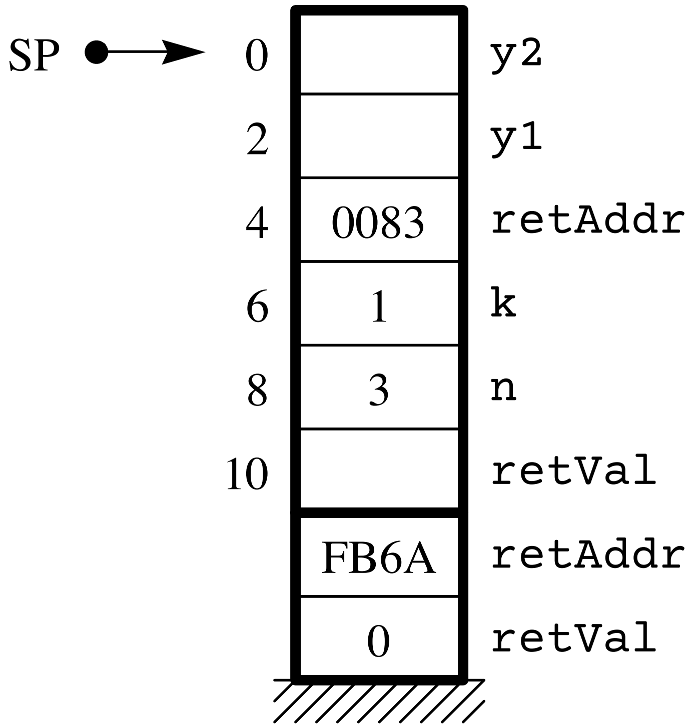
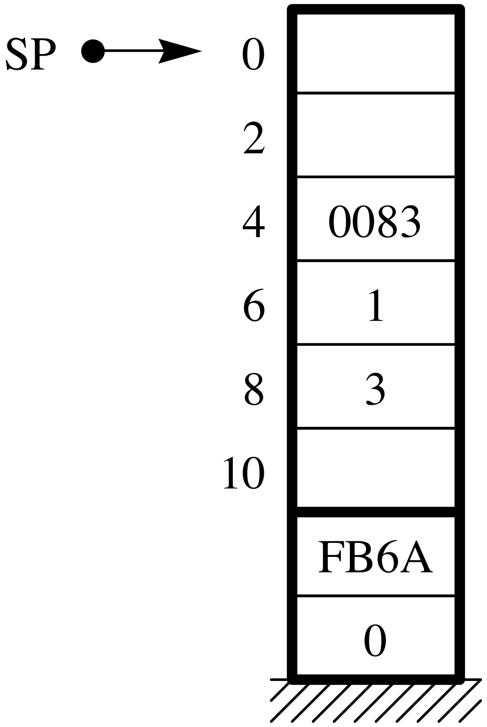


Figure 6.33

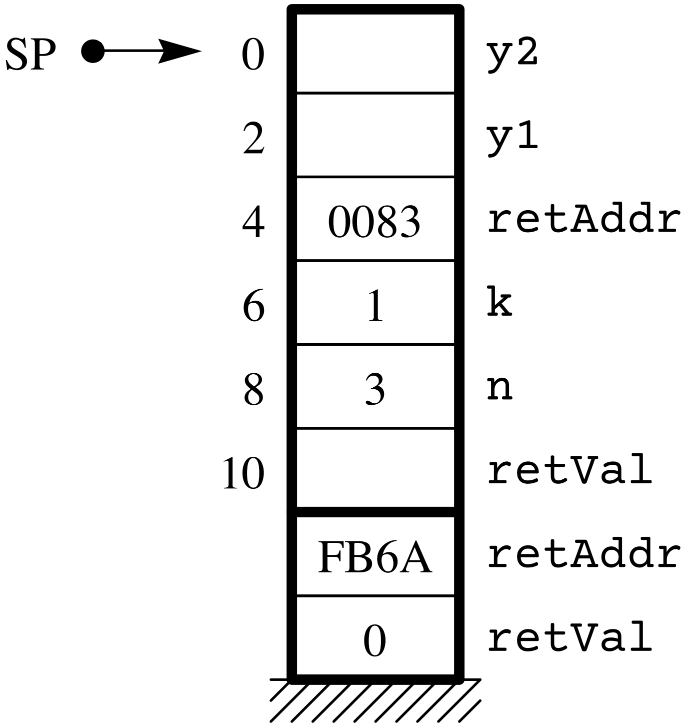
```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10          ;return value #2d
n:       .EQUATE 8           ;formal parameter #2d
k:       .EQUATE 6           ;formal parameter #2d
y1:      .EQUATE 2           ;local variable #2d
y2:      .EQUATE 0           ;local variable #2d
binCoeff:SUBSP   4,i         ;push #y1 #y2
if:      LDWA    k,s         ;if ((k == 0)
         BREQ    then
         LDWA    n,s         ;|| (n == k))
```



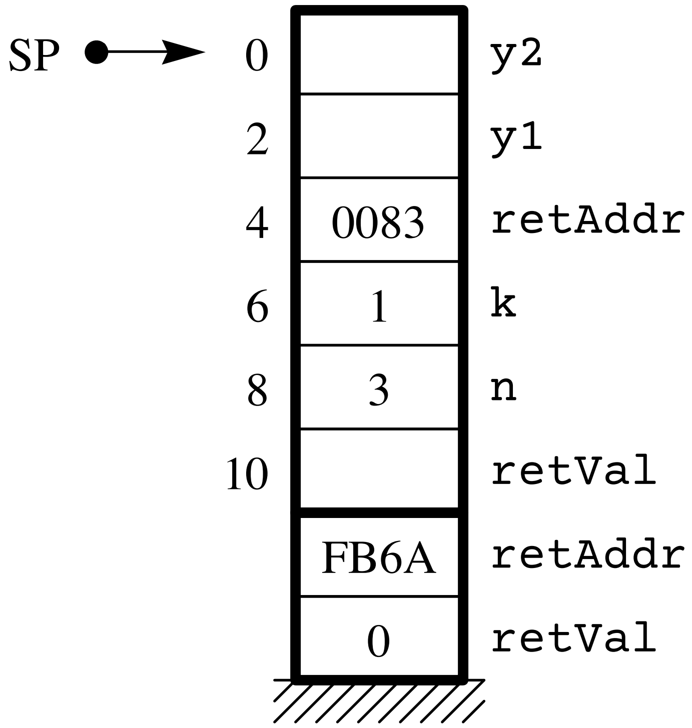
```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10           ;return value #2d
n:       .EQUATE 8            ;formal parameter #2d
k:       .EQUATE 6            ;formal parameter #2d
y1:      .EQUATE 2            ;local variable #2d
y2:      .EQUATE 0            ;local variable #2d
binCoeff:SUBSP   4,i          ;push #y1 #y2
if:      LDWA    k,s          ;if ((k == 0)
         BREQ    then
         LDWA    n,s          ;|| (n == k))
         CPWA    k,s
```



```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

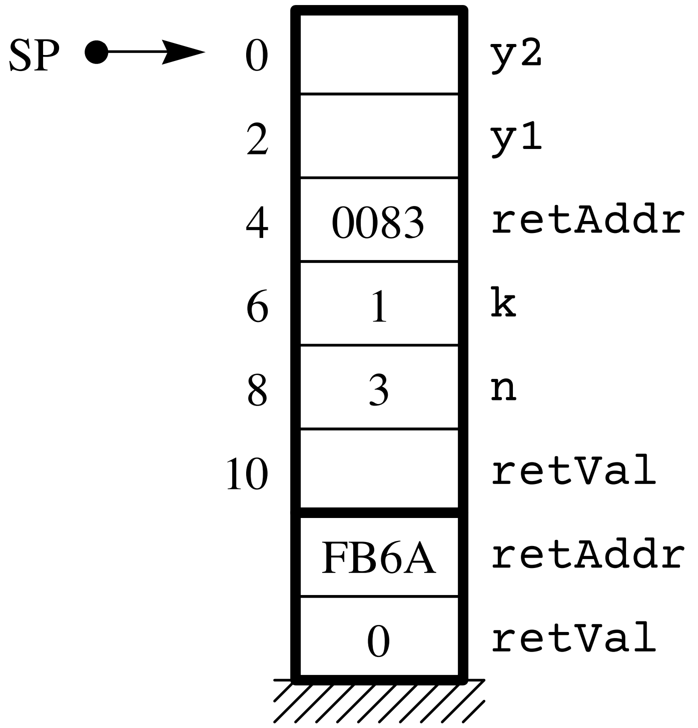
```
          BR        main
;
;******* int binCoeff(int n, int k)
retVal:   .EQUATE 10          ;return value #2d
n:        .EQUATE 8           ;formal parameter #2d
k:        .EQUATE 6           ;formal parameter #2d
y1:       .EQUATE 2           ;local variable #2d
y2:       .EQUATE 0           ;local variable #2d
binCoeff:SUBSP    4,i         ;push #y1 #y2
if:       LDWA    k,s         ;if ((k == 0)
          BREQ    then
          LDWA    n,s         ;|| (n == k))
          CPWA    k,s
          BRNE    else
```



0		y2
2		y1
4	0083	retAddr
6	1	k
8	3	n
10		retVal
	FB6A	retAddr
	0	retVal

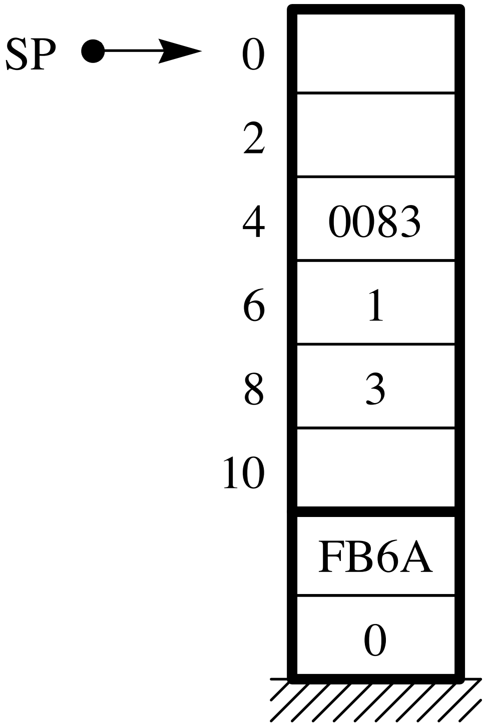
```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10           ;return value #2d
n:       .EQUATE 8            ;formal parameter #2d
k:       .EQUATE 6            ;formal parameter #2d
y1:      .EQUATE 2            ;local variable #2d
y2:      .EQUATE 0            ;local variable #2d
binCoeff:SUBSP   4,i          ;push #y1 #y2
if:      LDWA    k,s          ;if ((k == 0)
         BREQ    then
         LDWA    n,s          ;|| (n == k))
         CPWA    k,s
         BRNE    else
then:    LDWA    1,i          ;return 1
```



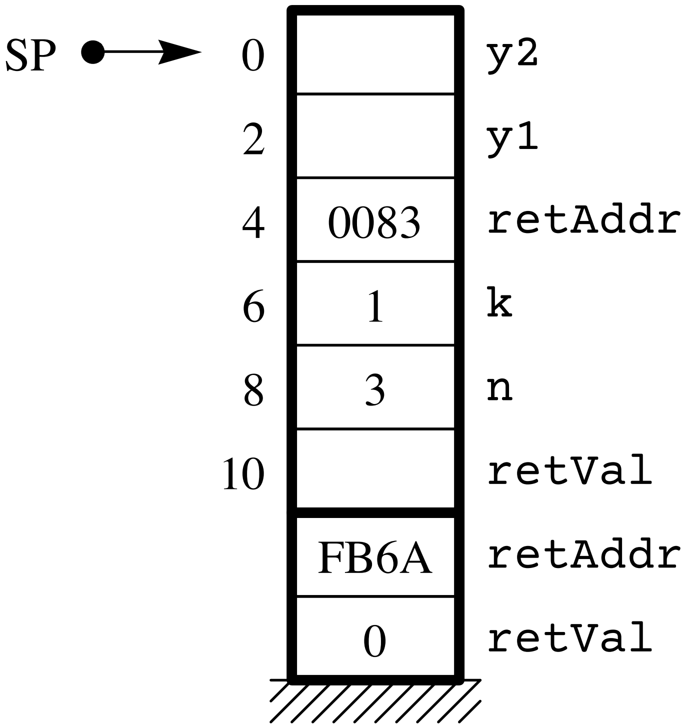
```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10         ;return value #2d
n:       .EQUATE 8          ;formal parameter #2d
k:       .EQUATE 6          ;formal parameter #2d
y1:      .EQUATE 2          ;local variable #2d
y2:      .EQUATE 0          ;local variable #2d
binCoeff:SUBSP   4,i        ;push #y1 #y2
if:      LDWA    k,s        ;if ((k == 0)
         BREQ    then
         LDWA    n,s        ;|| (n == k))
         CPWA    k,s
         BRNE    else
then:    LDWA    1,i        ;return 1
         STWA    retVal,s
```



```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf(”binCoeff(3, 1) = %d\n”,
        binCoeff(3, 1)); // ra1
    return 0;
}
```

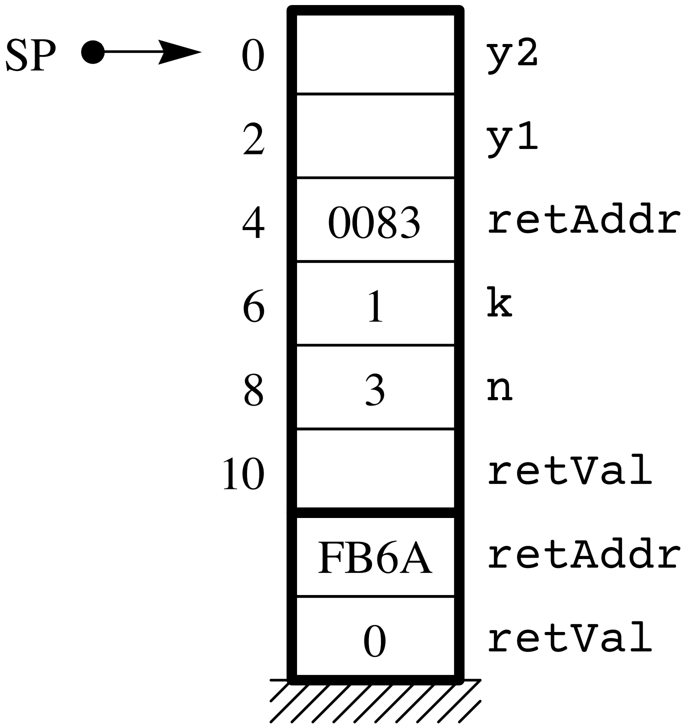
```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10          ;return value #2d
n:       .EQUATE 8           ;formal parameter #2d
k:       .EQUATE 6           ;formal parameter #2d
y1:      .EQUATE 2           ;local variable #2d
y2:      .EQUATE 0           ;local variable #2d
binCoeff:SUBSP   4,i         ;push #y1 #y2
if:      LDWA    k,s         ;if ((k == 0)
         BREQ    then
         LDWA    n,s         ;|| (n == k))
         CPWA    k,s
         BRNE    else
then:    LDWA    1,i         ;return 1
         STWA    retVal,s
         ADDSP   4,i         ;pop #y2 #y1
```



SP →	0		y2
	2		y1
	4	0083	retAddr
	6	1	k
	8	3	n
	10		retVal
		FB6A	retAddr
		0	retVal

```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10              ;return value #2d
n:       .EQUATE 8               ;formal parameter #2d
k:       .EQUATE 6               ;formal parameter #2d
y1:      .EQUATE 2               ;local variable #2d
y2:      .EQUATE 0               ;local variable #2d
binCoeff:SUBSP   4,i             ;push #y1 #y2
if:      LDWA    k,s             ;if ((k == 0)
         BREQ    then
         LDWA    n,s             ;|| (n == k))
         CPWA    k,s
         BRNE    else
then:    LDWA    1,i             ;return 1
         STWA    retVal,s
         ADDSP   4,i             ;pop #y2 #y1
         RET
```



```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10            ;return value #2d
n:       .EQUATE 8             ;formal parameter #2d
k:       .EQUATE 6             ;formal parameter #2d
y1:      .EQUATE 2             ;local variable #2d
y2:      .EQUATE 0             ;local variable #2d
binCoeff:SUBSP   4,i           ;push #y1 #y2
if:      LDWA    k,s           ;if ((k == 0)
         BREQ    then
         LDWA    n,s           ;|| (n == k))
         CPWA    k,s
         BRNE    else
then:    LDWA    1,i           ;return 1
         STWA    retVal,s
         ADDSP   4,i           ;pop #y2 #y1
         RET
```

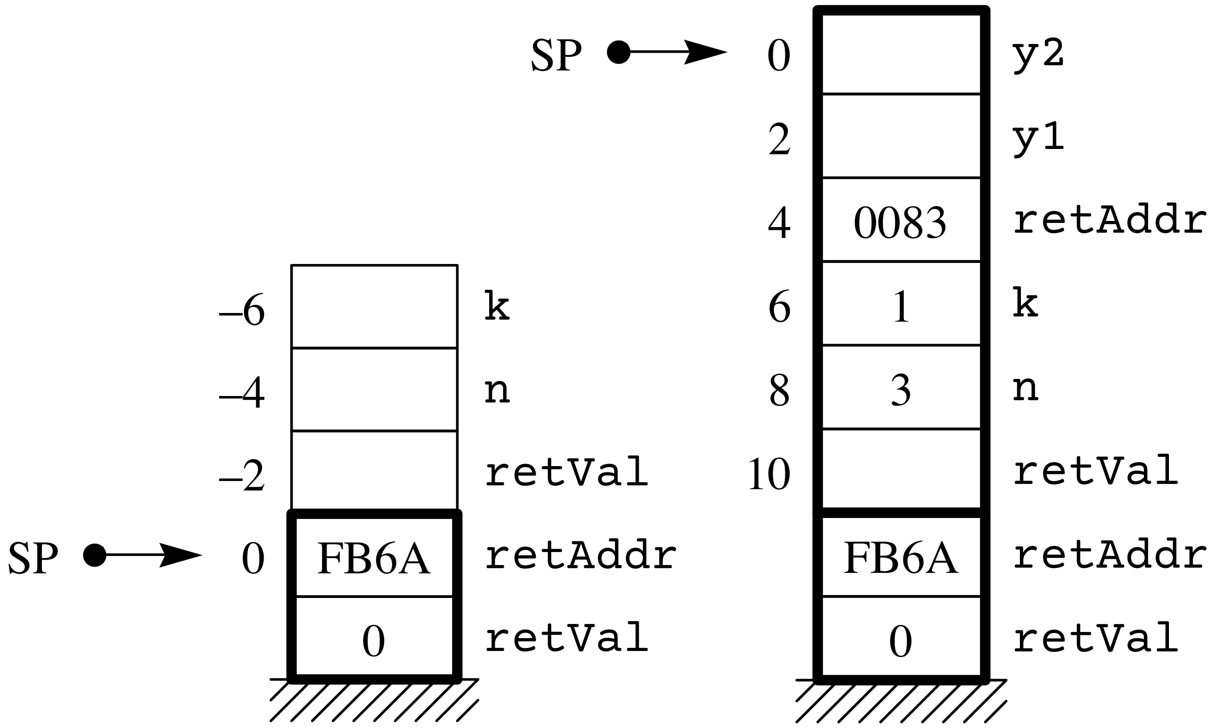


Figure 6.33

```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10          ;return value #2d
n:       .EQUATE 8           ;formal parameter #2d
k:       .EQUATE 6           ;formal parameter #2d
y1:      .EQUATE 2           ;local variable #2d
y2:      .EQUATE 0           ;local variable #2d
binCoeff:SUBSP   4,i         ;push #y1 #y2
if:      LDWA    k,s         ;if ((k == 0)
         BREQ    then
         LDWA    n,s         ;|| (n == k))
         CPWA    k,s
         BRNE    else
then:    LDWA    1,i         ;return 1
         STWA    retVal,s
         ADDSP   4,i         ;pop #y2 #y1
         RET
else:    LDWA    n,s         ;move n - 1
```

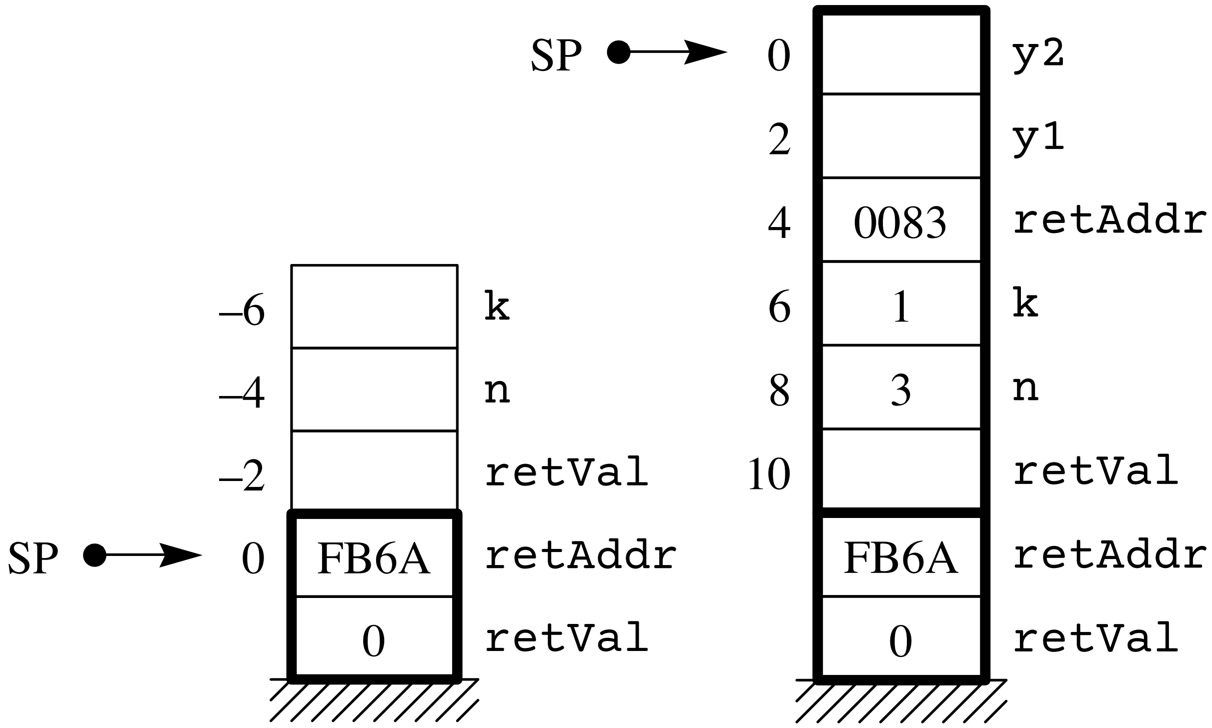
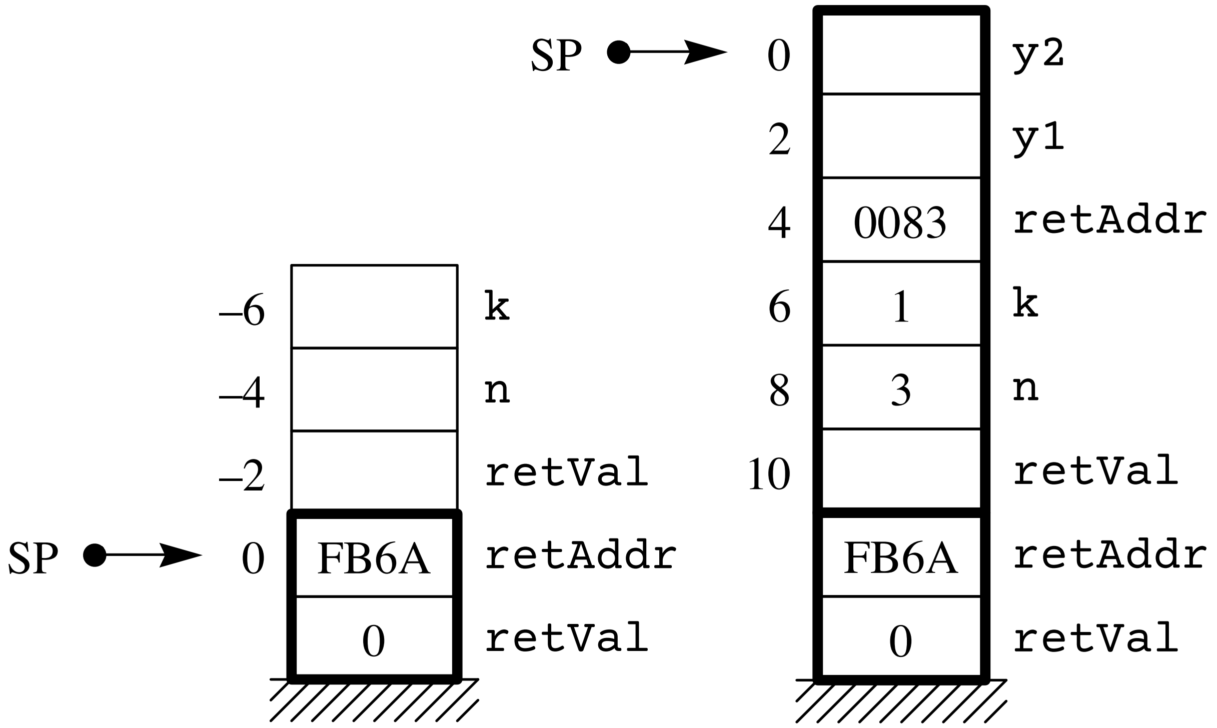


Figure 6.33

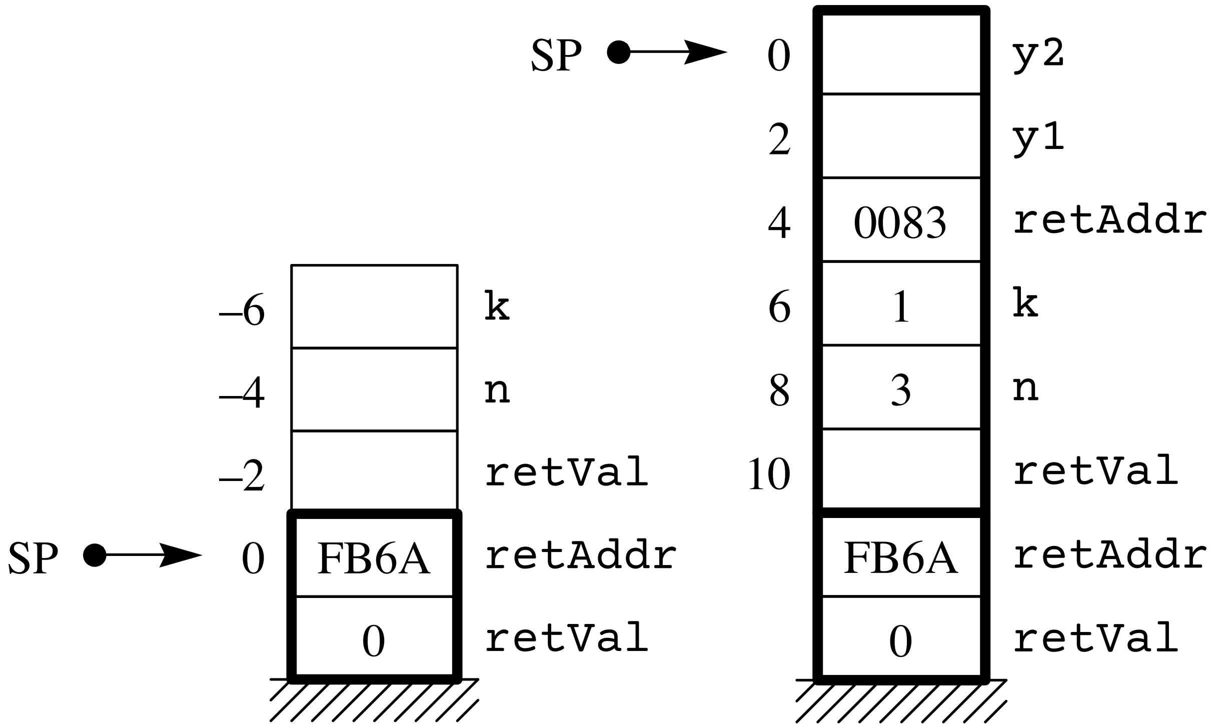
```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10          ;return value #2d
n:       .EQUATE 8           ;formal parameter #2d
k:       .EQUATE 6           ;formal parameter #2d
y1:      .EQUATE 2           ;local variable #2d
y2:      .EQUATE 0           ;local variable #2d
binCoeff:SUBSP   4,i         ;push #y1 #y2
if:      LDWA    k,s         ;if ((k == 0)
         BREQ    then
         LDWA    n,s         ;|| (n == k))
         CPWA    k,s
         BRNE    else
then:    LDWA    1,i         ;return 1
         STWA    retVal,s
         ADDSP   4,i         ;pop #y2 #y1
         RET
else:    LDWA    n,s         ;move n - 1
         SUBA    1,i
```



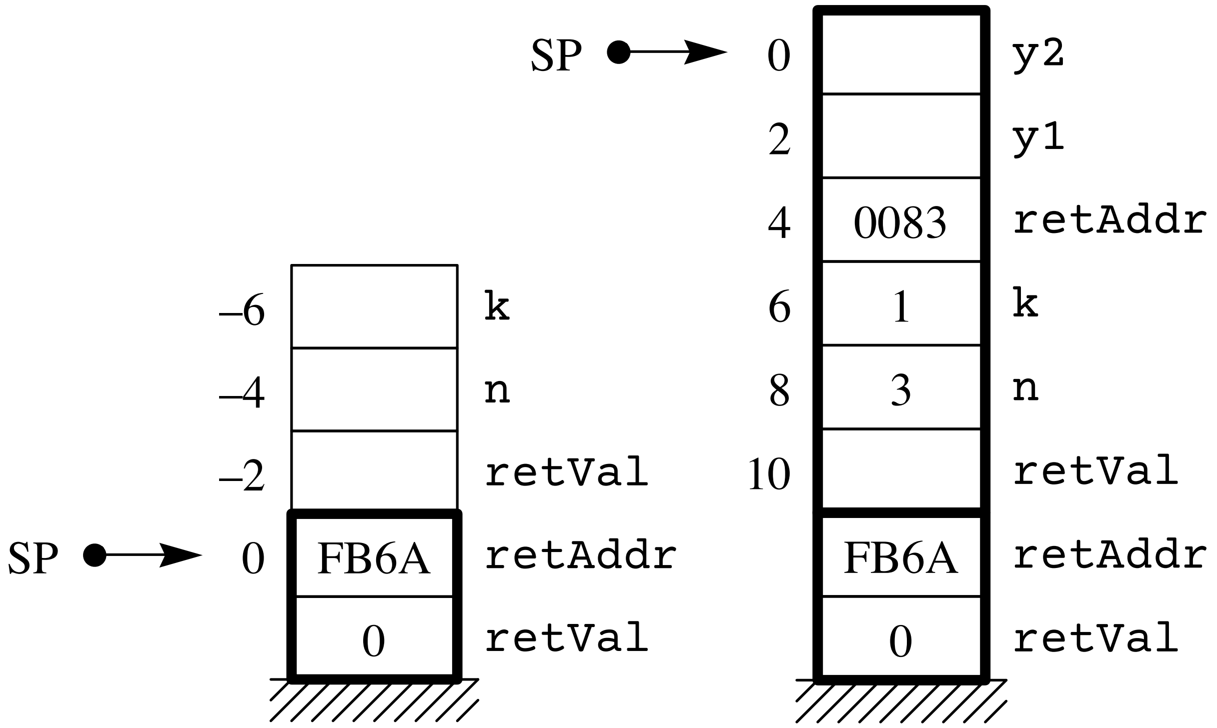
```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf(”binCoeff(3, 1) = %d\n”,
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10           ;return value #2d
n:       .EQUATE 8            ;formal parameter #2d
k:       .EQUATE 6            ;formal parameter #2d
y1:      .EQUATE 2            ;local variable #2d
y2:      .EQUATE 0            ;local variable #2d
binCoeff:SUBSP   4,i          ;push #y1 #y2
if:      LDWA    k,s          ;if ((k == 0)
         BREQ    then
         LDWA    n,s          ;|| (n == k))
         CPWA    k,s
         BRNE    else
then:    LDWA    1,i          ;return 1
         STWA    retVal,s
         ADDSP   4,i          ;pop #y2 #y1
         RET
else:    LDWA    n,s          ;move n - 1
         SUBA    1,i
         STWA    -4,s
```



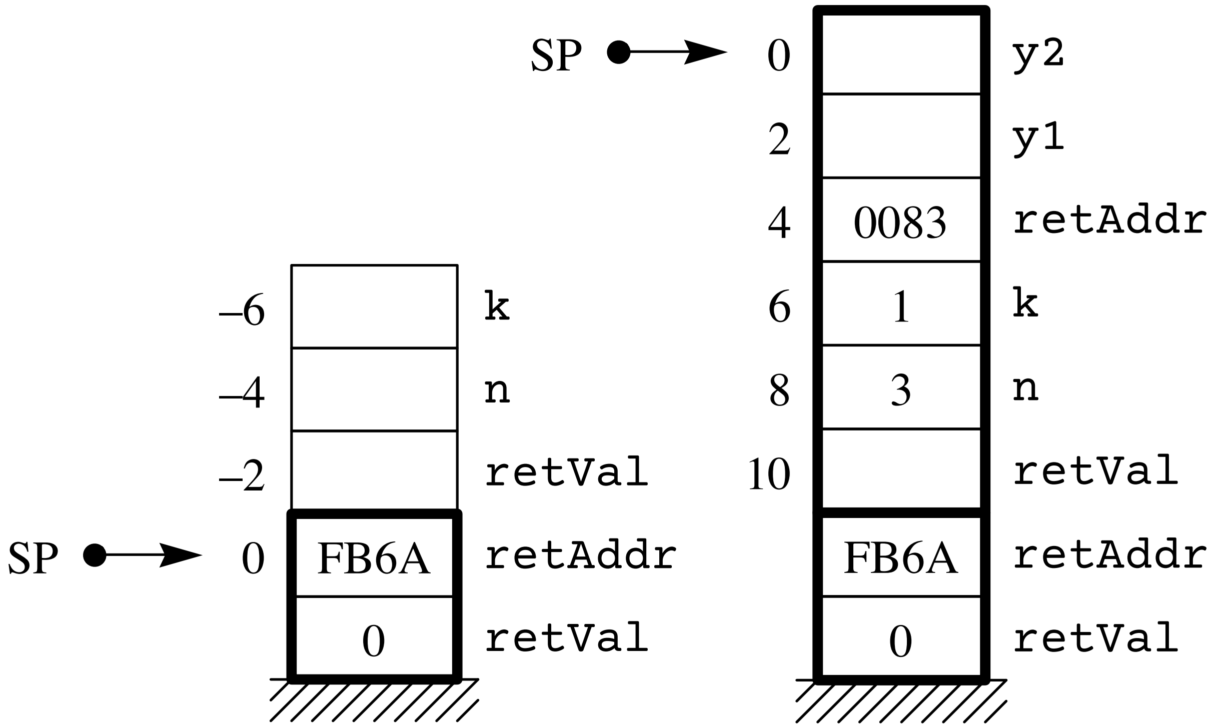
```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10           ;return value #2d
n:       .EQUATE 8            ;formal parameter #2d
k:       .EQUATE 6            ;formal parameter #2d
y1:      .EQUATE 2            ;local variable #2d
y2:      .EQUATE 0            ;local variable #2d
binCoeff:SUBSP   4,i          ;push #y1 #y2
if:      LDWA    k,s          ;if ((k == 0)
         BREQ    then
         LDWA    n,s          ;|| (n == k))
         CPWA    k,s
         BRNE    else
then:    LDWA    1,i          ;return 1
         STWA    retVal,s
         ADDSP   4,i          ;pop #y2 #y1
         RET
else:    LDWA    n,s          ;move n - 1
         SUBA    1,i
         STWA    -4,s
         LDWA    k,s          ;move k
```



```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
         BR       main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE  10          ;return value #2d
n:       .EQUATE  8           ;formal parameter #2d
k:       .EQUATE  6           ;formal parameter #2d
y1:      .EQUATE  2           ;local variable #2d
y2:      .EQUATE  0           ;local variable #2d
binCoeff:SUBSP    4,i         ;push #y1 #y2
if:      LDWA     k,s         ;if ((k == 0)
         BREQ     then
         LDWA     n,s         ;|| (n == k))
         CPWA     k,s
         BRNE     else
then:    LDWA     1,i         ;return 1
         STWA     retVal,s
         ADDSP    4,i         ;pop #y2 #y1
         RET
else:    LDWA     n,s         ;move n - 1
         SUBA     1,i
         STWA     -4,s
         LDWA     k,s         ;move k
         STWA     -6,s
```



```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```

```
          BR      main
;
;******* int binCoeff(int n, int k)
retVal:   .EQUATE 10          ;return value #2d
n:        .EQUATE 8           ;formal parameter #2d
k:        .EQUATE 6           ;formal parameter #2d
y1:       .EQUATE 2           ;local variable #2d
y2:       .EQUATE 0           ;local variable #2d
binCoeff:SUBSP    4,i         ;push #y1 #y2
if:       LDWA    k,s         ;if ((k == 0)
          BREQ    then
          LDWA    n,s         ;|| (n == k))
          CPWA    k,s
          BRNE    else
then:     LDWA    1,i         ;return 1
          STWA    retVal,s
          ADDSP   4,i         ;pop #y2 #y1
          RET
else:     LDWA    n,s         ;move n - 1
          SUBA    1,i
          STWA    -4,s
          LDWA    k,s         ;move k
          STWA    -6,s
          SUBSP   6,i         ;push #retVal #n #k
```

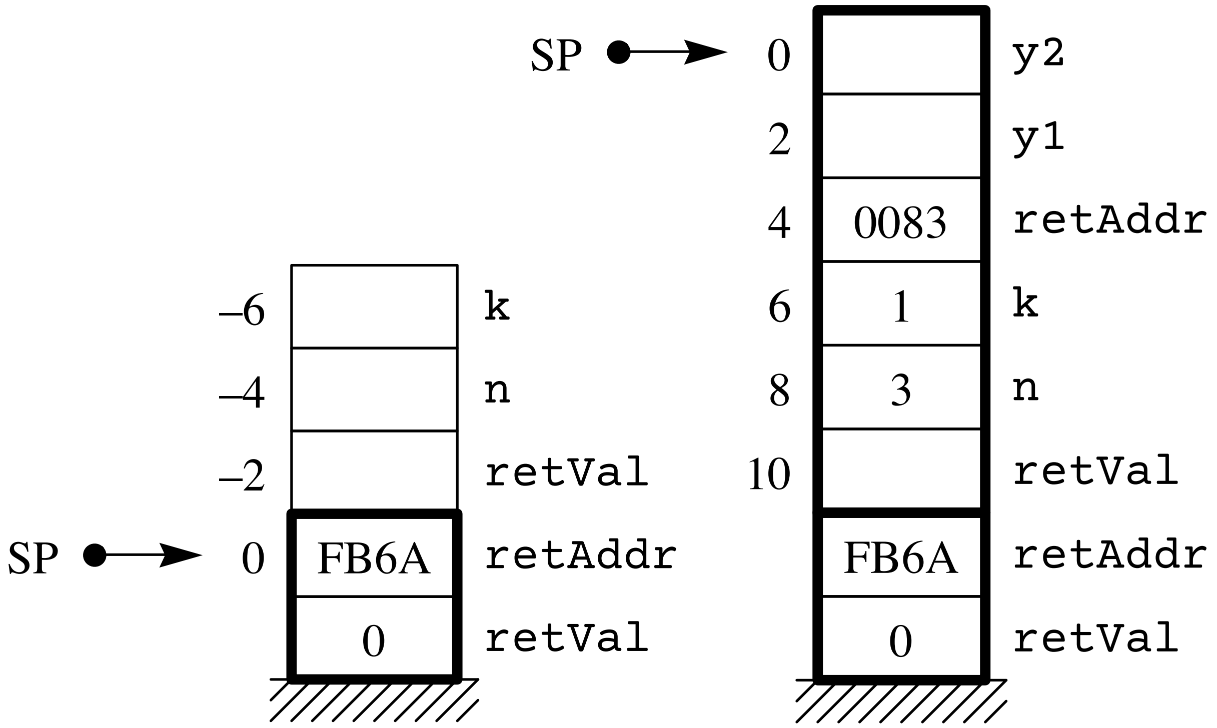
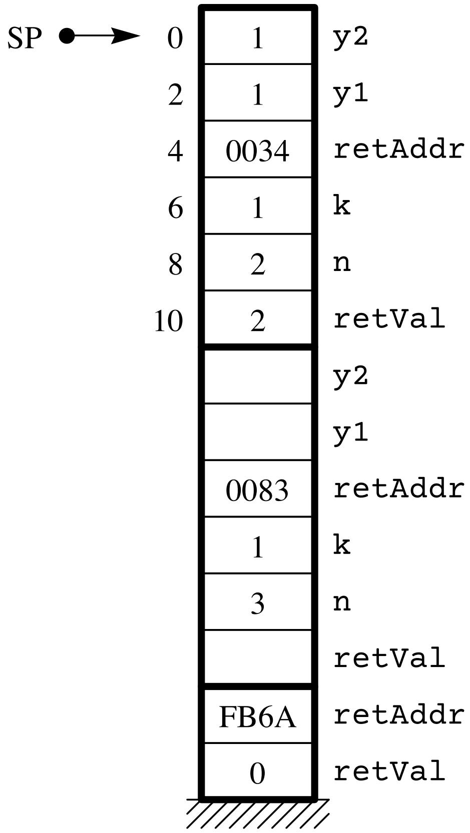


Figure 6.33

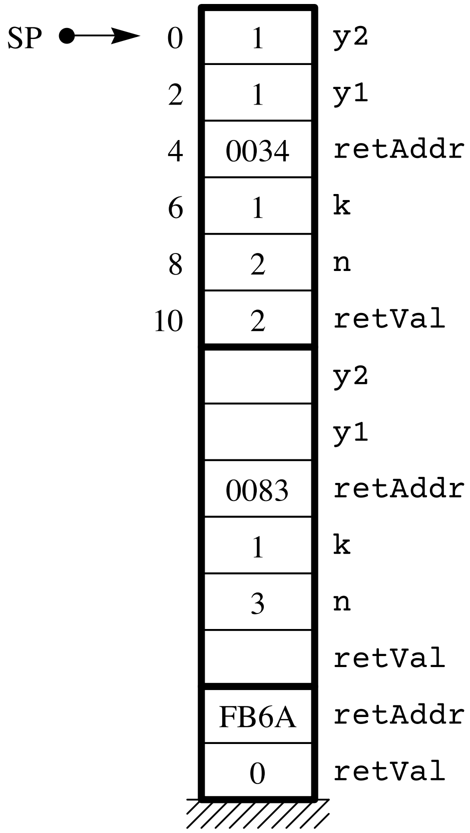
```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```



```
         BR       main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE  10          ;return value #2d
n:       .EQUATE  8           ;formal parameter #2d
k:       .EQUATE  6           ;formal parameter #2d
y1:      .EQUATE  2           ;local variable #2d
y2:      .EQUATE  0           ;local variable #2d
binCoeff:SUBSP    4,i         ;push #y1 #y2
if:      LDWA     k,s         ;if ((k == 0)
         BREQ     then
         LDWA     n,s         ;|| (n == k))
         CPWA     k,s
         BRNE     else
then:    LDWA     1,i         ;return 1
         STWA     retVal,s
         ADDSP    4,i         ;pop #y2 #y1
         RET
else:    LDWA     n,s         ;move n - 1
         SUBA     1,i
         STWA     -4,s
         LDWA     k,s         ;move k
         STWA     -6,s
         SUBSP    6,i         ;push #retVal #n #k
         CALL     binCoeff    ;binCoeff(n - 1, k)
```

Figure 6.33

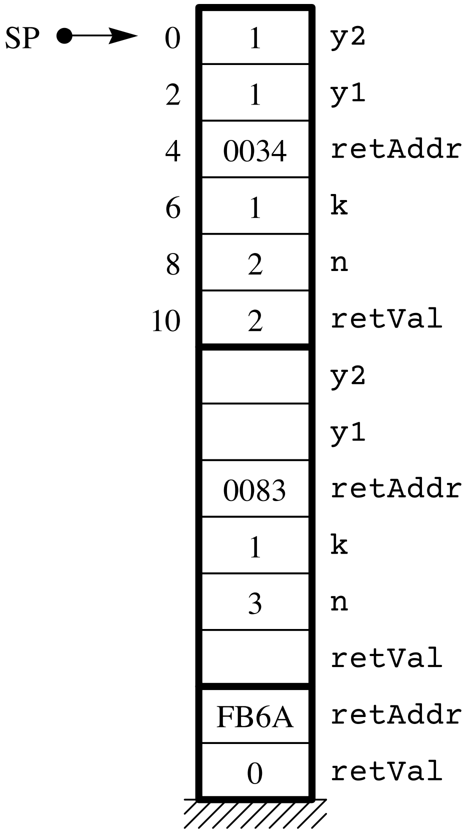
```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```



```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10          ;return value #2d
n:       .EQUATE 8           ;formal parameter #2d
k:       .EQUATE 6           ;formal parameter #2d
y1:      .EQUATE 2           ;local variable #2d
y2:      .EQUATE 0           ;local variable #2d
binCoeff:SUBSP   4,i         ;push #y1 #y2
if:      LDWA    k,s         ;if ((k == 0)
         BREQ    then
         LDWA    n,s         ;|| (n == k))
         CPWA    k,s
         BRNE    else
then:    LDWA    1,i         ;return 1
         STWA    retVal,s
         ADDSP   4,i         ;pop #y2 #y1
         RET
else:    LDWA    n,s         ;move n - 1
         SUBA    1,i
         STWA    -4,s
         LDWA    k,s         ;move k
         STWA    -6,s
         SUBSP   6,i         ;push #retVal #n #k
         CALL    binCoeff    ;binCoeff(n - 1, k)
ra2:     ADDSP   6,i         ;pop #k #n #retVal
```

Figure 6.33

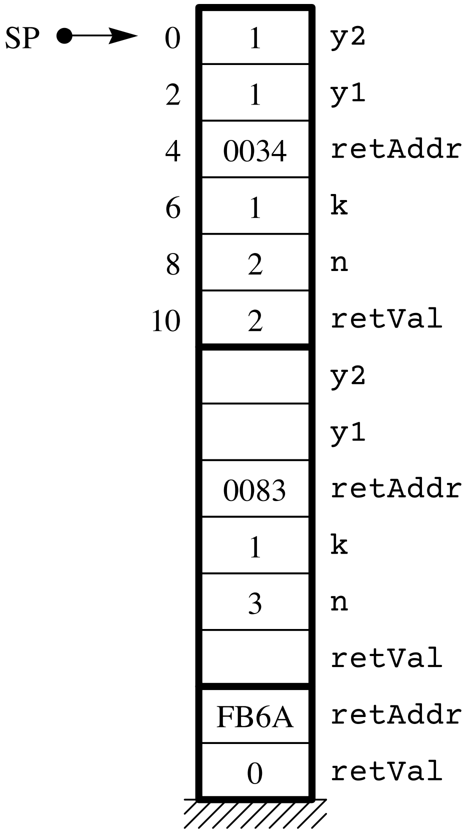
```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```



SP →	0	1	y2
	2	1	y1
	4	0034	retAddr
	6	1	k
	8	2	n
	10	2	retVal
			y2
			y1
		0083	retAddr
		1	k
		3	n
			retVal
		FB6A	retAddr
		0	retVal

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10            ;return value #2d
n:       .EQUATE 8             ;formal parameter #2d
k:       .EQUATE 6             ;formal parameter #2d
y1:      .EQUATE 2             ;local variable #2d
y2:      .EQUATE 0             ;local variable #2d
binCoeff:SUBSP   4,i           ;push #y1 #y2
if:      LDWA    k,s           ;if ((k == 0)
         BREQ    then
         LDWA    n,s           ;|| (n == k))
         CPWA    k,s
         BRNE    else
then:    LDWA    1,i           ;return 1
         STWA    retVal,s
         ADDSP   4,i           ;pop #y2 #y1
         RET
else:    LDWA    n,s           ;move n - 1
         SUBA    1,i
         STWA    -4,s
         LDWA    k,s           ;move k
         STWA    -6,s
         SUBSP   6,i           ;push #retVal #n #k
         CALL    binCoeff      ;binCoeff(n - 1, k)
ra2:     ADDSP   6,i           ;pop #k #n #retVal
         LDWA    -2,s          ;y1 = binCoeff(n - 1, k)
```

```
#include <stdio.h>
int binCoeff(int n, int k) {
    int y1, y2;
    if ((k == 0) || (n == k)) {
        return 1;
    } else {
        y1 = binCoeff(n - 1, k); // ra2
        y2 = binCoeff(n - 1, k - 1); // ra3
        return y1 + y2;
    }
}

int main() {
    printf("binCoeff(3, 1) = %d\n",
        binCoeff(3, 1)); // ra1
    return 0;
}
```



SP			
SP ●→	0	1	y2
	2	1	y1
	4	0034	retAddr
	6	1	k
	8	2	n
	10	2	retVal
			y2
			y1
		0083	retAddr
		1	k
		3	n
			retVal
		FB6A	retAddr
		0	retVal

```
         BR      main
;
;******* int binCoeff(int n, int k)
retVal:  .EQUATE 10          ;return value #2d
n:       .EQUATE 8           ;formal parameter #2d
k:       .EQUATE 6           ;formal parameter #2d
y1:      .EQUATE 2           ;local variable #2d
y2:      .EQUATE 0           ;local variable #2d
binCoeff:SUBSP   4,i         ;push #y1 #y2
if:      LDWA    k,s         ;if ((k == 0)
         BREQ    then
         LDWA    n,s         ;|| (n == k))
         CPWA    k,s
         BRNE    else
then:    LDWA    1,i         ;return 1
         STWA    retVal,s
         ADDSP   4,i         ;pop #y2 #y1
         RET
else:    LDWA    n,s         ;move n - 1
         SUBA    1,i
         STWA    -4,s
         LDWA    k,s         ;move k
         STWA    -6,s
         SUBSP   6,i         ;push #retVal #n #k
         CALL    binCoeff    ;binCoeff(n - 1, k)
ra2:     ADDSP   6,i         ;pop #k #n #retVal
         LDWA    -2,s        ;y1 = binCoeff(n - 1, k)
         STWA    y1,s
```

Immediate addressing

- Oprnd = OprndSpec
- Asmb5 letter: `i`
- The operand specifier *is* the operand.

Direct addressing

- Oprnd = Mem[OprndSpec]
- Asmb5 letter: `d`
- The operand specifier is the *address* in memory of the operand.

Stack-relative addressing

- Oprnd = Mem[SP + OprndSpec]
- Asmb5 letter: s
- The stack pointer *plus* the operand specifier is the *address* in memory of the operand.

Stack-relative deferred addressing

- Oprnd = Mem[Mem[SP + OprndSpec]]
- Asmb5 letters: `sf`
- The stack pointer *plus* the operand specifier is the *address* in memory of the *address* in memory of the operand.

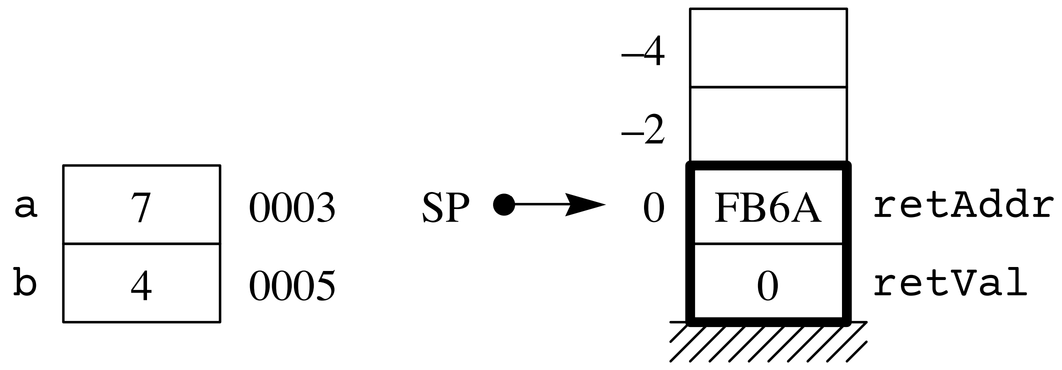
Translating global pointer parameters

- To access the actual parameter in the caller, generate load with immediate addressing (`i`)
- To access the formal parameter `x` in the callee, generate load with stack-relative addressing (`s`)
- To access the dereferenced formal parameter `*x` in the callee, generate load with stack-relative deferred addressing (`sf`)

Figure 6.35

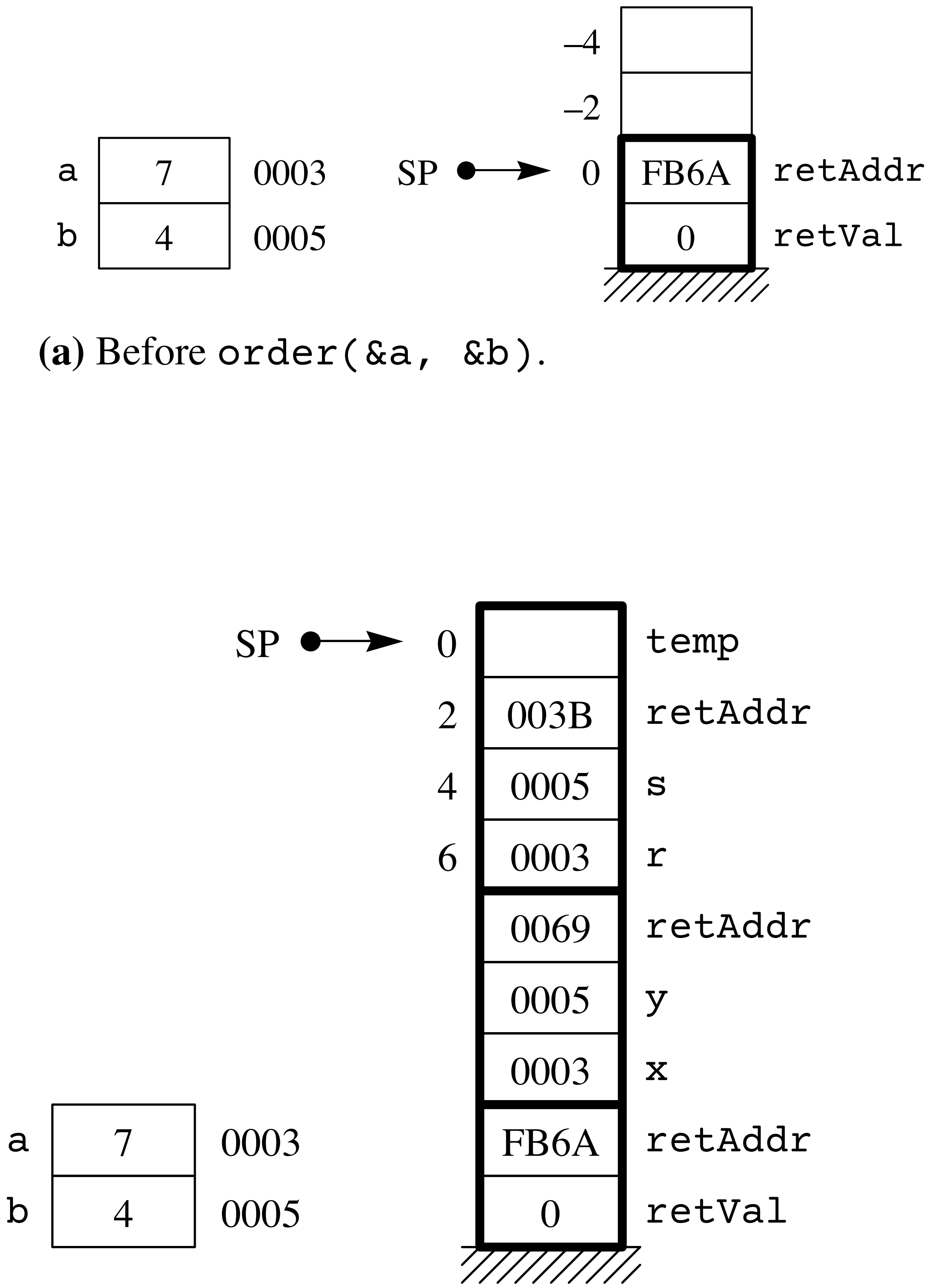
```
int a, b;

void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}

int main() {
    printf(”Enter an integer: ”);
    scanf(”%d”, &a);
    printf(”Enter an integer: ”);
    scanf(”%d”, &b);
    order(&a, &b);
    printf(”Ordered they are: %d, %d\n”,
        a, b); // ra1
    return 0;
}
```

```
int a, b;

void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}

int main() {
    printf(”Enter an integer: ”);
    scanf(”%d”, &a);
    printf(”Enter an integer: ”);
    scanf(”%d”, &b);
    order(&a, &b);
    printf(”Ordered they are: %d, %d\n”,
        a, b); // ra1
    return 0;
}
```



(**a**) Before `order(&a, &b)`.

```
int a, b;

void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}

int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```



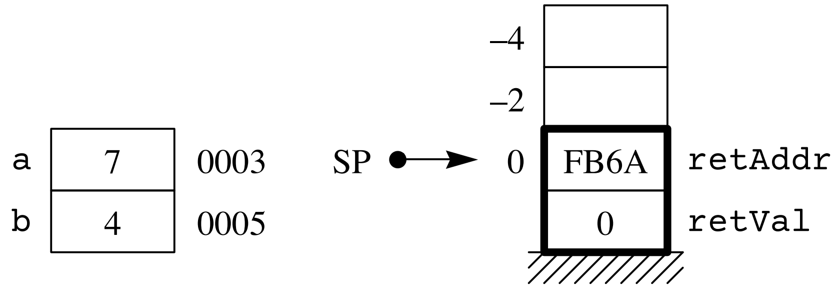
(a) Before `order(&a, &b)`.

(b) After `swap(x, y)`.

```
                BR      main
```

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

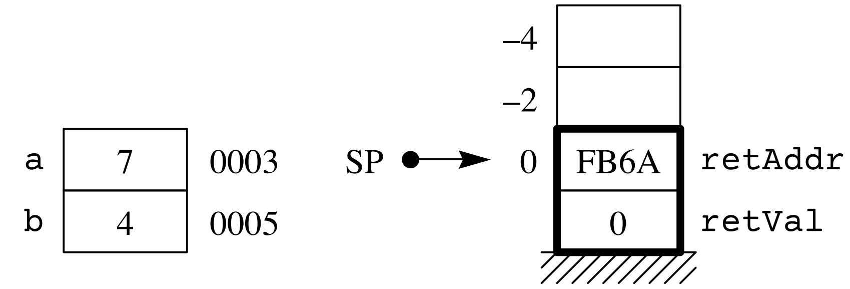
```
;
;******* main()
```



(a) Before `order(&a, &b)`.

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

```
          BR      main
a:        .BLOCK  2            ;global variable #2d







;
;******* main()
```

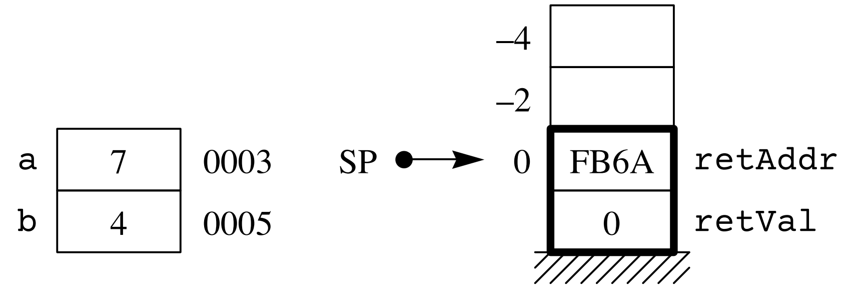


(a) Before `order(&a, &b)`.

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

```
         BR      main
a:       .BLOCK  2          ;global variable #2d
b:       .BLOCK  2          ;global variable #2d
```

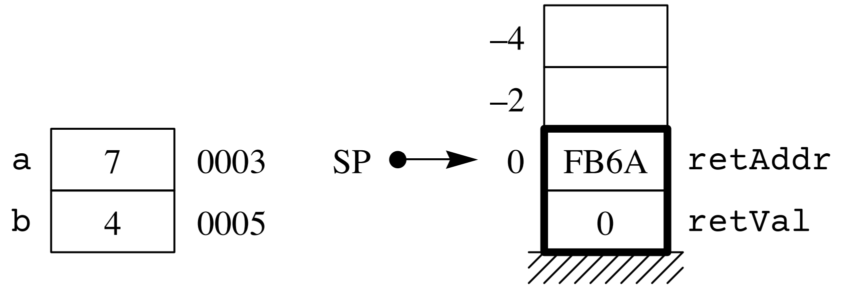
```
;
;******* main()
```



(a) Before order(&a, &b).

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

```
                BR        main
   a:           .BLOCK    2            ;global variable #2d
   b:           .BLOCK    2            ;global variable #2d

;
;******* main()
main:      @STRO      msg1,d       ;printf("Enter an integer: ")
```



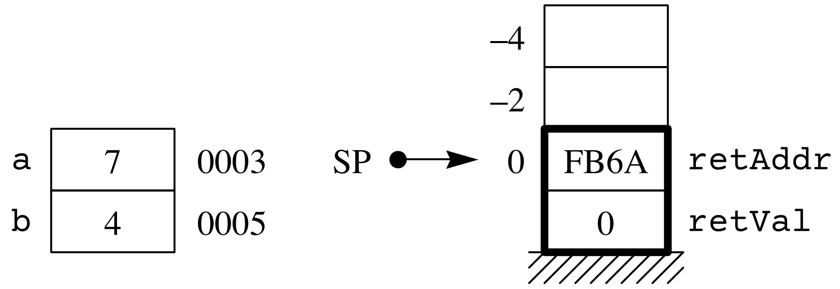
(a) Before `order(&a, &b)`.

Figure 6.35

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

```
             BR        main
a:           .BLOCK    2              ;global variable #2d
b:           .BLOCK    2              ;global variable #2d
```

```
;
;******* main()
main:    @STRO     msg1,d        ;printf("Enter an integer: ")
         @DECI     a,d           ;scanf("%d", &a)
```

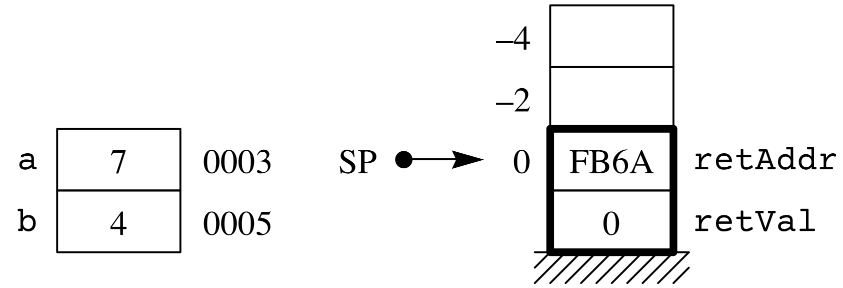


(a) Before `order(&a, &b)`.

Figure 6.35

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

```
          BR        main
a:        .BLOCK    2              ;global variable #2d
b:        .BLOCK    2              ;global variable #2d
;
;******* main()
main:     @STRO     msg1,d         ;printf("Enter an integer: ")
          @DECI     a,d            ;scanf("%d", &a)
          @STRO     msg1,d         ;printf("Enter an integer: ")
```

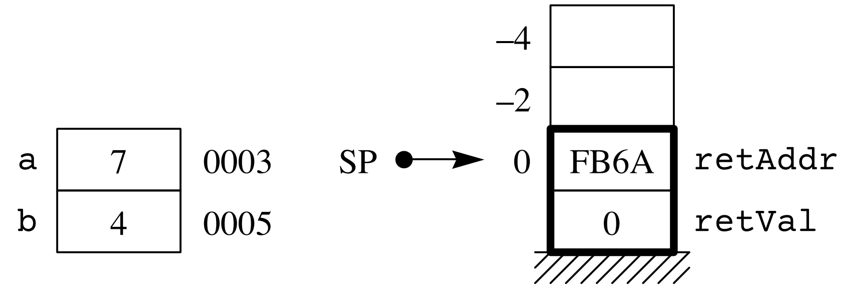


(a) Before order(&a, &b).

Computer Systems Sixth edition Figure 6.35

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

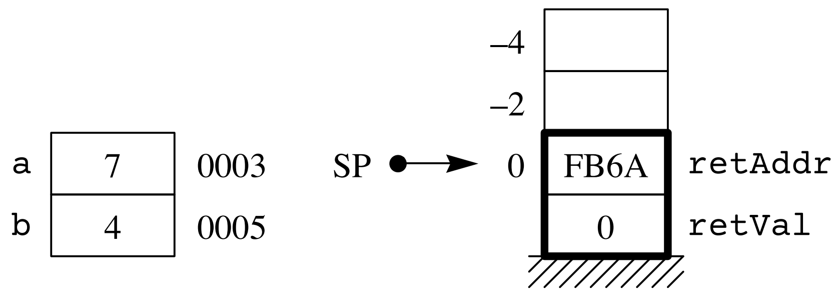
```
          BR        main
a:        .BLOCK    2             ;global variable #2d
b:        .BLOCK    2             ;global variable #2d





;
;******* main()
main:     @STRO     msg1,d        ;printf("Enter an integer: ")
          @DECI     a,d           ;scanf("%d", &a)
          @STRO     msg1,d        ;printf("Enter an integer: ")
          @DECI     b,d           ;scanf("%d", &b)
```



(a) Before order(&a, &b).

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

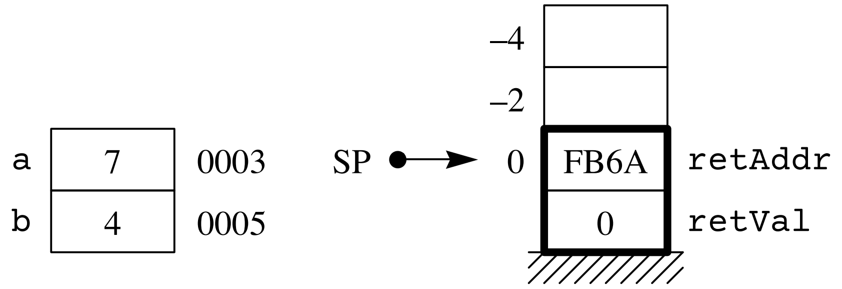
```
              BR        main
a:            .BLOCK    2             ;global variable #2d
b:            .BLOCK    2             ;global variable #2d

;
;******* main()
main:         @STRO     msg1,d        ;printf("Enter an integer: ")
              @DECI     a,d           ;scanf("%d", &a)
              @STRO     msg1,d        ;printf("Enter an integer: ")
              @DECI     b,d           ;scanf("%d", &b)
              LDWA     a,i            ;move &a
```



(a) Before `order(&a, &b)`.

```
int main() {
    printf(”Enter an integer: ”);
    scanf(”%d”, &a);
    printf(”Enter an integer: ”);
    scanf(”%d”, &b);
    order(&a, &b);
    printf(”Ordered they are: %d, %d\n”,
        a, b); // ra1
    return 0;
}
```

```
                BR         main
a:              .BLOCK     2               ;global variable #2d
b:              .BLOCK     2               ;global variable #2d

;
;******* main()
main:          @STRO      msg1,d         ;printf(”Enter an integer: ”)
               @DECI      a,d            ;scanf(”%d”, &a)
               @STRO      msg1,d         ;printf(”Enter an integer: ”)
               @DECI      b,d            ;scanf(”%d”, &b)
               LDWA      a,i             ;move &a
               STWA      -2,s
```

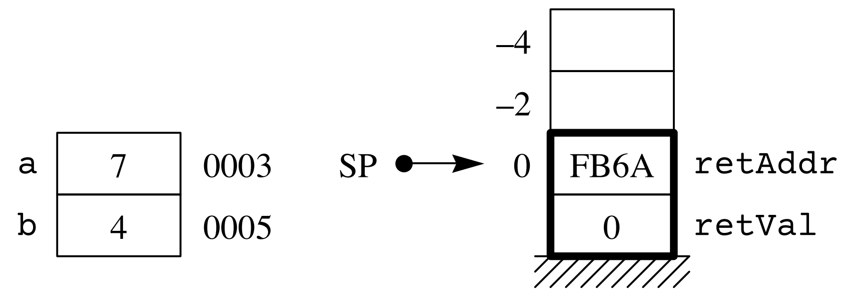


(a) Before `order(&a, &b)`.

Figure 6.35

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

```
         BR        main
a:       .BLOCK    2             ;global variable #2d
b:       .BLOCK    2             ;global variable #2d

;
;******* main()
main:    @STRO     msg1,d        ;printf("Enter an integer: ")
         @DECI     a,d           ;scanf("%d", &a)
         @STRO     msg1,d        ;printf("Enter an integer: ")
         @DECI     b,d           ;scanf("%d", &b)
         LDWA      a,i           ;move &a
         STWA      -2,s
         LDWA      b,i           ;move &b
```

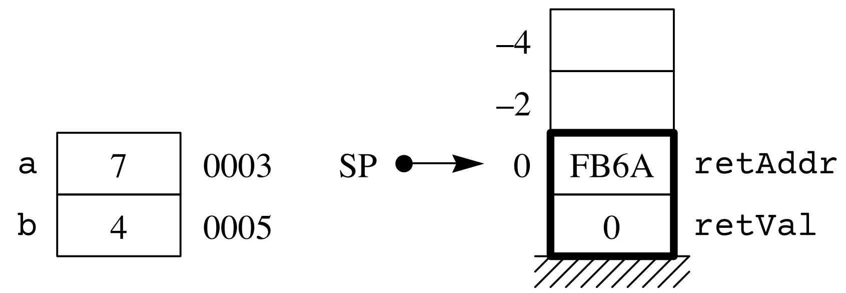


(a) Before `order(&a, &b)`.

Computer Systems Sixth edition Figure 6.35

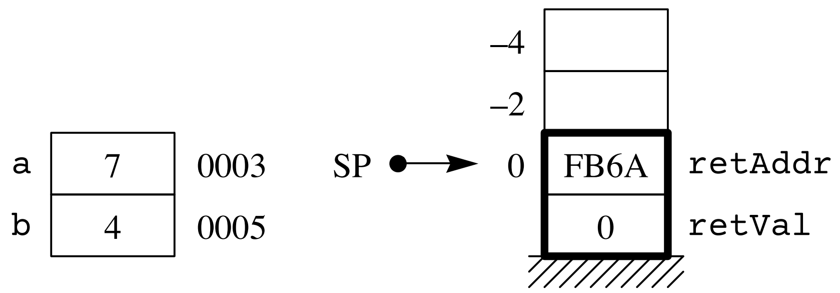
```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

```
         BR       main
a:       .BLOCK   2             ;global variable #2d
b:       .BLOCK   2             ;global variable #2d

;
;******* main()
main:    @STRO    msg1,d        ;printf("Enter an integer: ")
         @DECI    a,d           ;scanf("%d", &a)
         @STRO    msg1,d        ;printf("Enter an integer: ")
         @DECI    b,d           ;scanf("%d", &b)
         LDWA     a,i           ;move &a
         STWA     -2,s
         LDWA     b,i           ;move &b
         STWA     -4,s
```



(a) Before `order(&a, &b).`

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

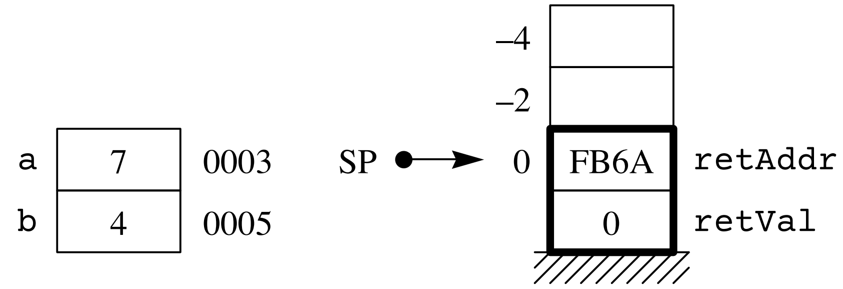


(a) Before `order(&a, &b)`.

```
         BR      main
a:       .BLOCK  2            ;global variable #2d
b:       .BLOCK  2            ;global variable #2d
```

```
;
;******* main()
main:    @STRO   msg1,d       ;printf("Enter an integer: ")
         @DECI   a,d          ;scanf("%d", &a)
         @STRO   msg1,d       ;printf("Enter an integer: ")
         @DECI   b,d          ;scanf("%d", &b)
         LDWA    a,i          ;move &a
         STWA    -2,s
         LDWA    b,i          ;move &b
         STWA    -4,s
         SUBSP   4,i          ;push #x #y
```

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```



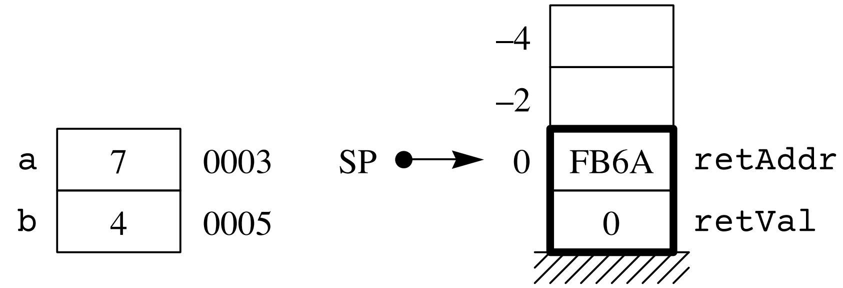
(a) Before `order(&a, &b)`.

```
         BR      main
a:       .BLOCK  2              ;global variable #2d
b:       .BLOCK  2              ;global variable #2d





;
;******* main()
main:    @STRO    msg1,d        ;printf("Enter an integer: ")
         @DECI    a,d           ;scanf("%d", &a)
         @STRO    msg1,d        ;printf("Enter an integer: ")
         @DECI    b,d           ;scanf("%d", &b)
         LDWA    a,i            ;move &a
         STWA    -2,s
         LDWA    b,i            ;move &b
         STWA    -4,s
         SUBSP   4,i            ;push #x #y
         CALL    order          ;order(&a, &b)
```

Figure 6.35

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

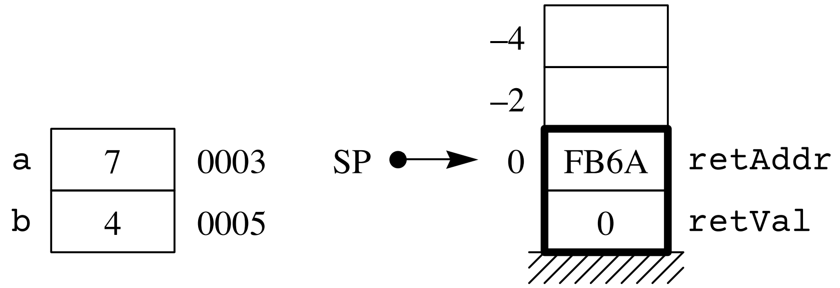
```
                 BR        main
   a:            .BLOCK    2              ;global variable #2d
   b:            .BLOCK    2              ;global variable #2d




 ;
 ;******* main()
 main:        @STRO     msg1,d          ;printf("Enter an integer: ")
              @DECI     a,d             ;scanf("%d", &a)
              @STRO     msg1,d          ;printf("Enter an integer: ")
              @DECI     b,d             ;scanf("%d", &b)
              LDWA     a,i              ;move &a
              STWA     -2,s
              LDWA     b,i              ;move &b
              STWA     -4,s
              SUBSP    4,i              ;push #x #y
              CALL     order            ;order(&a, &b)
 ra1:         ADDSP    4,i              ;pop #y #x
```



(a) Before `order(&a, &b)`.

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

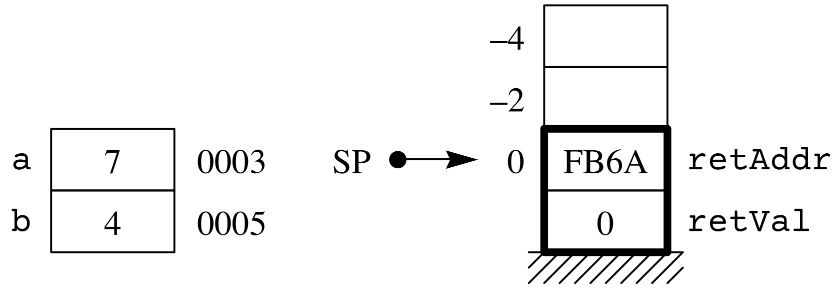
```
         BR      main
a:       .BLOCK  2            ;global variable #2d
b:       .BLOCK  2            ;global variable #2d

;
;******* main()
main:    @STRO   msg1,d       ;printf("Enter an integer: ")
         @DECI   a,d          ;scanf("%d", &a)
         @STRO   msg1,d       ;printf("Enter an integer: ")
         @DECI   b,d          ;scanf("%d", &b)
         LDWA    a,i          ;move &a
         STWA    -2,s
         LDWA    b,i          ;move &b
         STWA    -4,s
         SUBSP   4,i          ;push #x #y
         CALL    order        ;order(&a, &b)
ra1:     ADDSP   4,i          ;pop #y #x
         @STRO   msg2,d       ;printf("Ordered they are: %d
```



(**a**) Before order(&a, &b).

Figure 6.35

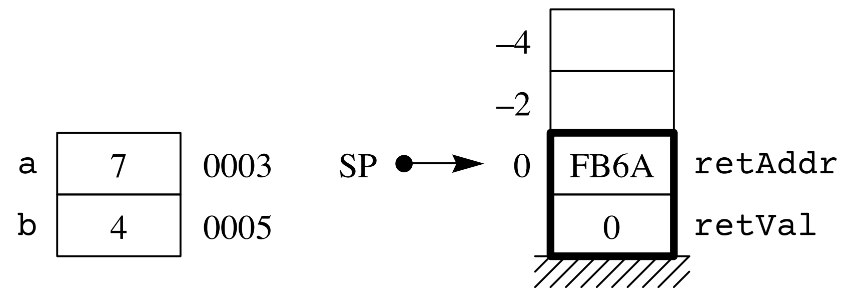
```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```



(a) Before `order(&a, &b)`.

```
         BR       main
a:       .BLOCK   2           ;global variable #2d
b:       .BLOCK   2           ;global variable #2d

;
;******* main()
main:    @STRO     msg1,d     ;printf("Enter an integer: ")
         @DECI     a,d        ;scanf("%d", &a)
         @STRO     msg1,d     ;printf("Enter an integer: ")
         @DECI     b,d        ;scanf("%d", &b)
         LDWA     a,i         ;move &a
         STWA     -2,s
         LDWA     b,i         ;move &b
         STWA     -4,s
         SUBSP    4,i         ;push #x #y
         CALL     order       ;order(&a, &b)
ra1:     ADDSP    4,i         ;pop #y #x
         @STRO     msg2,d     ;printf("Ordered they are: %d
         @DECO     a,d        ;, a
```

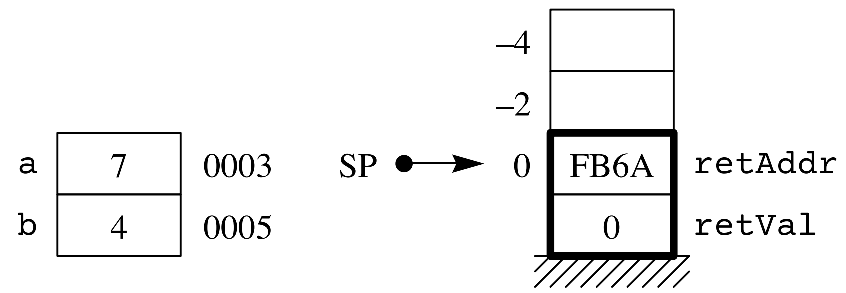
```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```



(a) Before `order(&a, &b)`.

```
            BR        main
a:          .BLOCK    2            ;global variable #2d
b:          .BLOCK    2            ;global variable #2d




;
;******* main()
main:    @STRO     msg1,d      ;printf("Enter an integer: ")
         @DECI     a,d         ;scanf("%d", &a)
         @STRO     msg1,d      ;printf("Enter an integer: ")
         @DECI     b,d         ;scanf("%d", &b)
         LDWA     a,i          ;move &a
         STWA     -2,s
         LDWA     b,i          ;move &b
         STWA     -4,s
         SUBSP    4,i          ;push #x #y
         CALL     order        ;order(&a, &b)
ra1:     ADDSP    4,i          ;pop #y #x
         @STRO     msg2,d      ;printf("Ordered they are: %d
         @DECO     a,d         ;, a
         @STRO     msg3,d
```

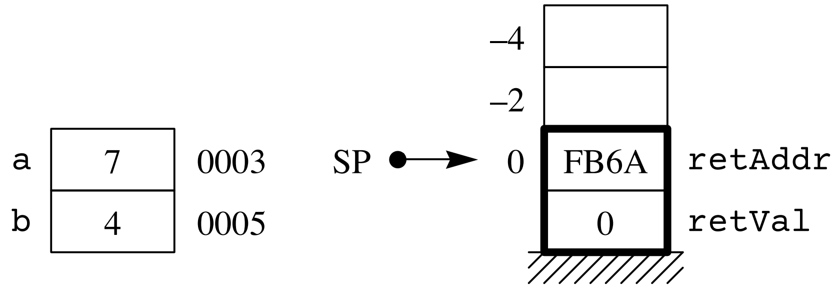
```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```



(a) Before `order(&a, &b)`.

```
         BR      main
a:       .BLOCK  2              ;global variable #2d
b:       .BLOCK  2              ;global variable #2d
;
;******* main()
main:    @STRO    msg1,d        ;printf("Enter an integer: ")
         @DECI    a,d           ;scanf("%d", &a)
         @STRO    msg1,d        ;printf("Enter an integer: ")
         @DECI    b,d           ;scanf("%d", &b)
         LDWA    a,i            ;move &a
         STWA    -2,s
         LDWA    b,i            ;move &b
         STWA    -4,s
         SUBSP   4,i            ;push #x #y
         CALL    order          ;order(&a, &b)
ra1:     ADDSP   4,i            ;pop #y #x
         @STRO    msg2,d        ;printf("Ordered they are: %d
         @DECO    a,d           ;, a
         @STRO    msg3,d
         @DECO    b,d           ;, b)
```

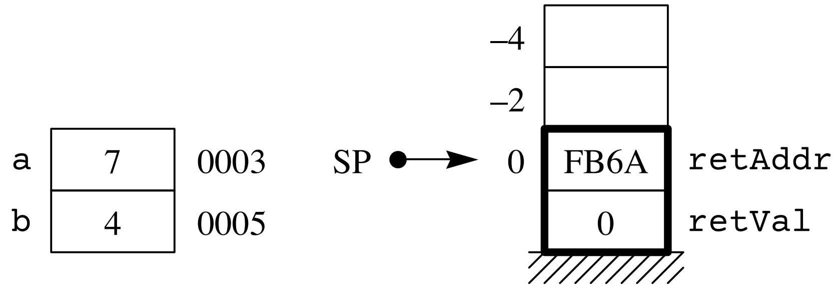
```
int main() {
    printf(”Enter an integer: ”);
    scanf(”%d”, &a);
    printf(”Enter an integer: ”);
    scanf(”%d”, &b);
    order(&a, &b);
    printf(”Ordered they are: %d, %d\n”,
        a, b); // ra1
    return 0;
}
```



(a) Before `order(&a, &b)`.

```
         BR       main
a:       .BLOCK   2             ;global variable #2d
b:       .BLOCK   2             ;global variable #2d

;
;******* main()
main:    @STRO     msg1,d       ;printf(”Enter an integer: ”)
         @DECI     a,d          ;scanf(”%d”, &a)
         @STRO     msg1,d       ;printf(”Enter an integer: ”)
         @DECI     b,d          ;scanf(”%d”, &b)
         LDWA     a,i           ;move &a
         STWA     -2,s
         LDWA     b,i           ;move &b
         STWA     -4,s
         SUBSP    4,i           ;push #x #y
         CALL     order         ;order(&a, &b)
ra1:     ADDSP    4,i           ;pop #y #x
         @STRO     msg2,d       ;printf(”Ordered they are: %d
         @DECO     a,d          ;, a
         @STRO     msg3,d
         @DECO     b,d          ;, b)
         @CHARO    ’\n’,i
```

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

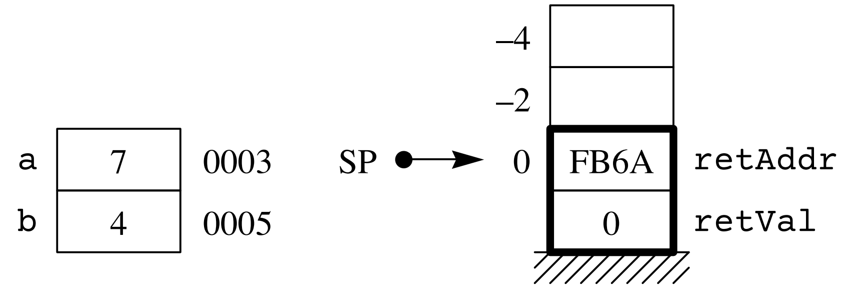


(a) Before `order(&a, &b)`.

```
         BR        main
a:       .BLOCK    2             ;global variable #2d
b:       .BLOCK    2             ;global variable #2d





;
;******* main()
main:    @STRO     msg1,d       ;printf("Enter an integer: ")
         @DECI     a,d          ;scanf("%d", &a)
         @STRO     msg1,d       ;printf("Enter an integer: ")
         @DECI     b,d          ;scanf("%d", &b)
         LDWA     a,i          ;move &a
         STWA     -2,s
         LDWA     b,i          ;move &b
         STWA     -4,s
         SUBSP    4,i          ;push #x #y
         CALL     order        ;order(&a, &b)
ra1:     ADDSP    4,i          ;pop #y #x
         @STRO     msg2,d       ;printf("Ordered they are: %d
         @DECO     a,d          ;, a
         @STRO     msg3,d
         @DECO     b,d          ;, b)
         @CHARO    '\n',i
         RET
```

```
int main() {
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

```
         BR       main
a:       .BLOCK   2             ;global variable #2d
b:       .BLOCK   2             ;global variable #2d
```



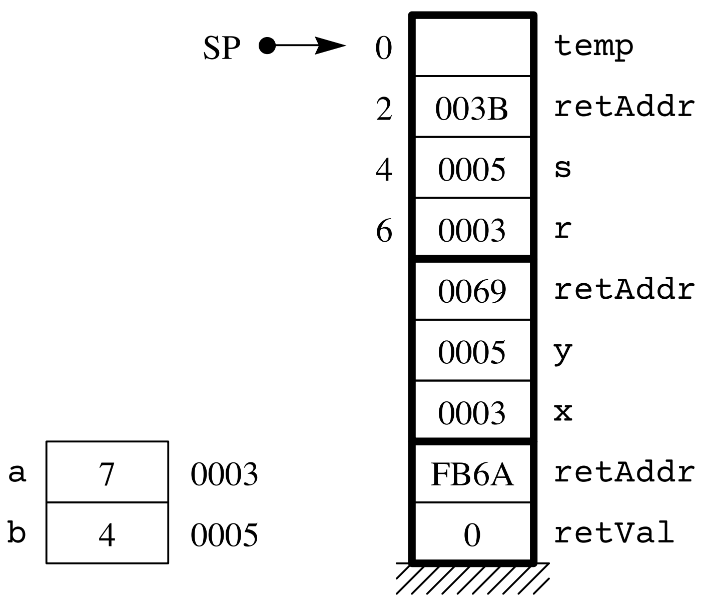
(a) Before `order(&a, &b)`.

```
;
;******* main()
main:    @STRO    msg1,d        ;printf("Enter an integer: ")
         @DECI    a,d           ;scanf("%d", &a)
         @STRO    msg1,d        ;printf("Enter an integer: ")
         @DECI    b,d           ;scanf("%d", &b)
         LDWA     a,i           ;move &a
         STWA     -2,s
         LDWA     b,i           ;move &b
         STWA     -4,s
         SUBSP    4,i           ;push #x #y
         CALL     order         ;order(&a, &b)
ra1:     ADDSP    4,i           ;pop #y #x
         @STRO    msg2,d        ;printf("Ordered they are: %d
         @DECO    a,d           ;, a
         @STRO    msg3,d
         @DECO    b,d           ;, b)
         @CHARO   '\n',i
         RET
msg1:    .ASCII   "Enter an integer: \0"
msg2:    .ASCII   "Ordered they are: \0"
msg3:    .ASCII   ", \0"
```

Figure 6.35

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

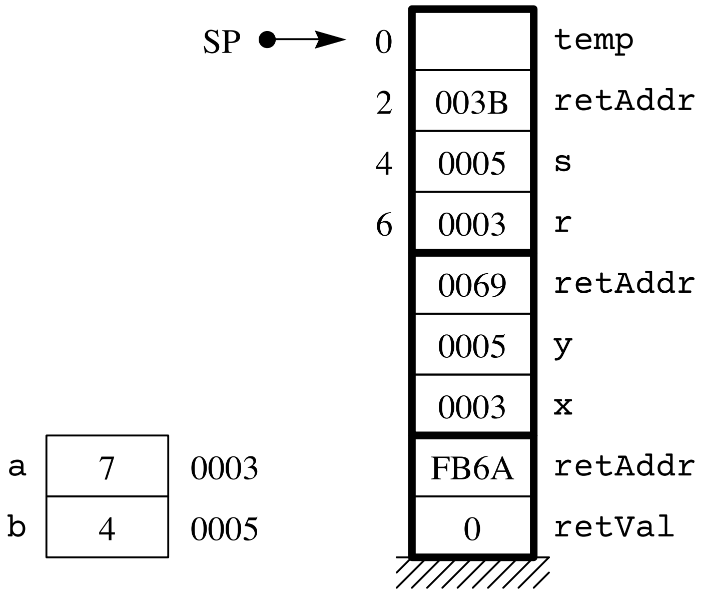
```
;******* void order(int *x, int *y)
```



(b) After `swap(x, y)`.

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

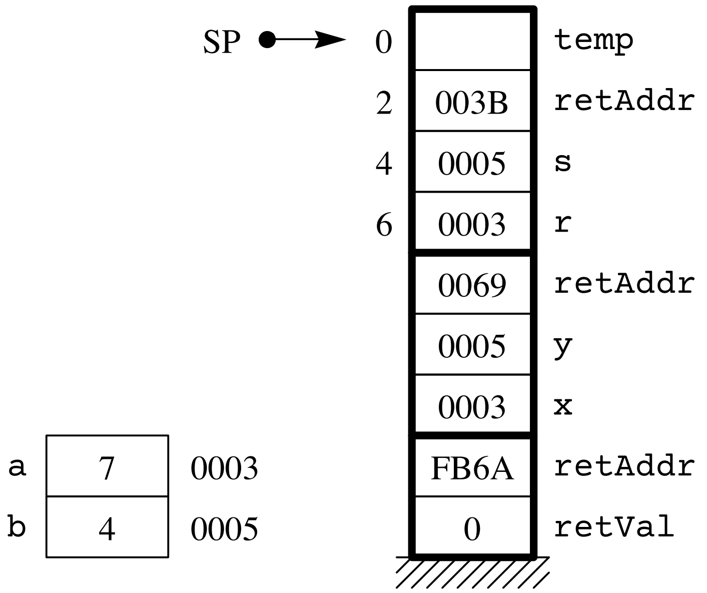
```
;******* void order(int *x, int *y)
x:       .EQUATE 4           ;formal parameter #2h
```



(b) After `swap(x, y)`.

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

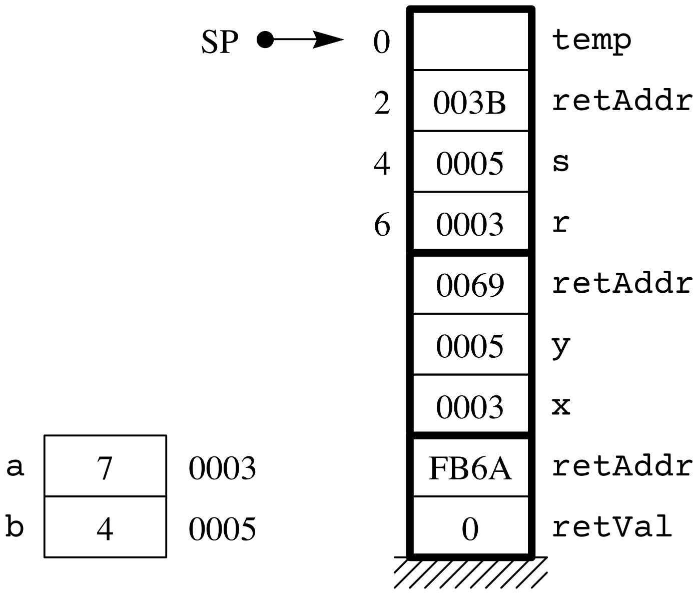
```
;******* void order(int *x, int *y)
x:       .EQUATE 4             ;formal parameter #2h
y:       .EQUATE 2             ;formal parameter #2h
```



(b) After `swap(x, y)`.

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

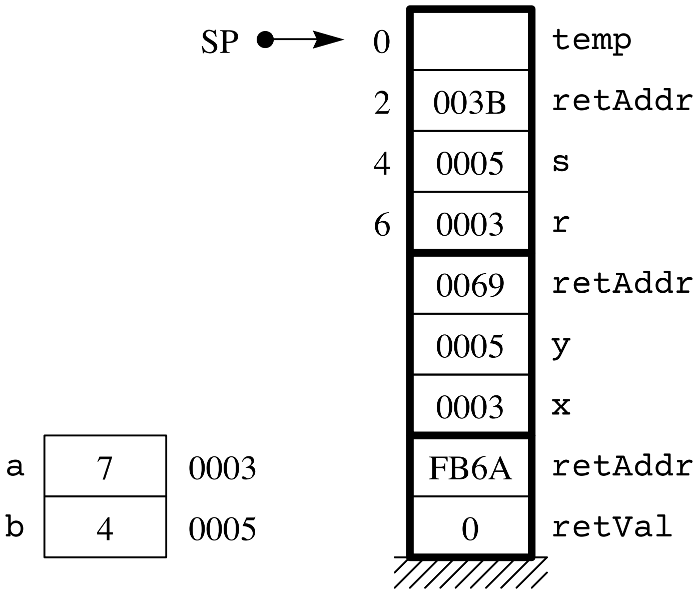
```
;******* void order(int *x, int *y)
x:       .EQUATE 4              ;formal parameter #2h
y:       .EQUATE 2              ;formal parameter #2h
order:   LDWA    x,sf           ;if (*x > *y)
```



(b) After swap(x, y).

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

```
;******* void order(int *x, int *y)
x:       .EQUATE 4           ;formal parameter #2h
y:       .EQUATE 2           ;formal parameter #2h
order:   LDWA    x,sf        ;if (*x > *y)
         CPWA    y,sf
```



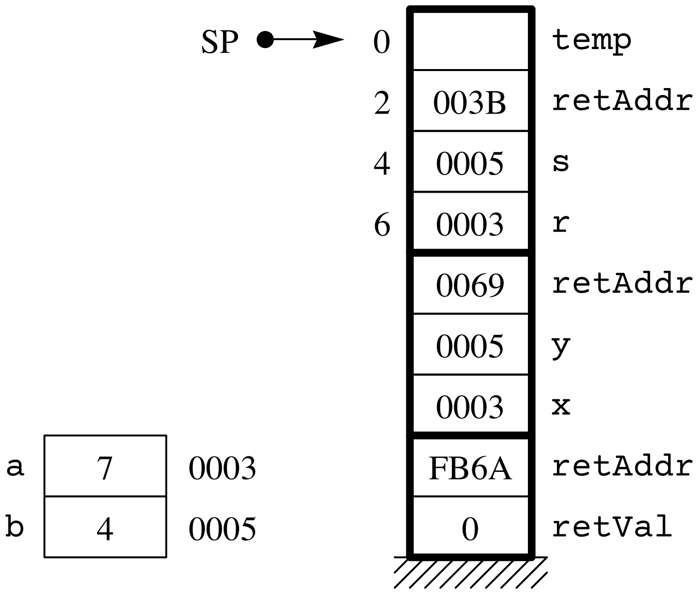
(b) After swap(x, y).

Computer Systems Sixth edition

Figure 6.35

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

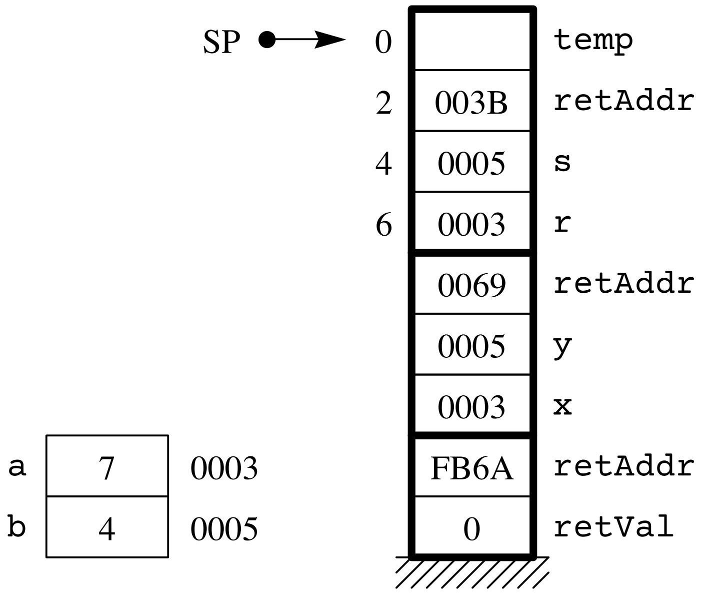
```
;******* void order(int *x, int *y)
x:       .EQUATE 4             ;formal parameter #2h
y:       .EQUATE 2             ;formal parameter #2h
order:   LDWA    x,sf          ;if (*x > *y)
         CPWA    y,sf
         BRLE    endIf
```



(b) After swap(x, y).

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

```
;******* void order(int *x, int *y)
x:        .EQUATE 4             ;formal parameter #2h
y:        .EQUATE 2             ;formal parameter #2h
order:    LDWA    x,sf          ;if (*x > *y)
          CPWA    y,sf
          BRLE    endIf
          LDWA    x,s           ;move x
```

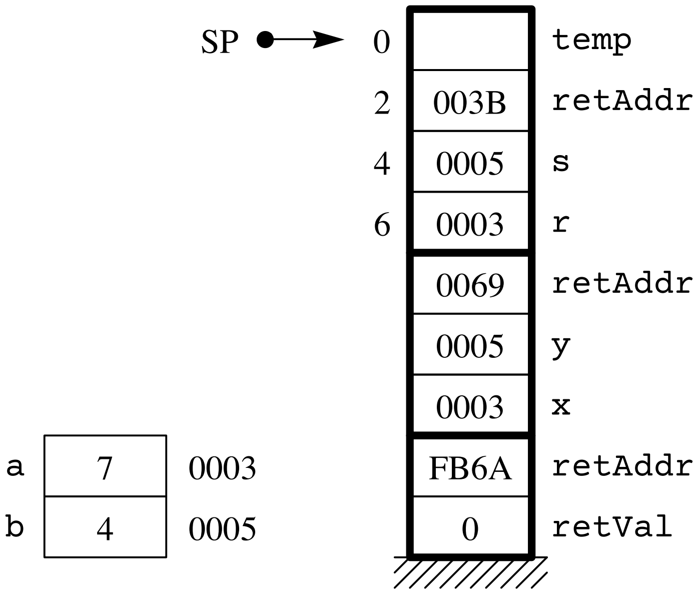


(b) After swap(x, y).

Figure 6.35

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

```
;******* void order(int *x, int *y)
x:         .EQUATE 4             ;formal parameter #2h
y:         .EQUATE 2             ;formal parameter #2h
order:     LDWA    x,sf          ;if (*x > *y)
           CPWA    y,sf
           BRLE    endIf
           LDWA    x,s           ;move x
           STWA    -2,s
```

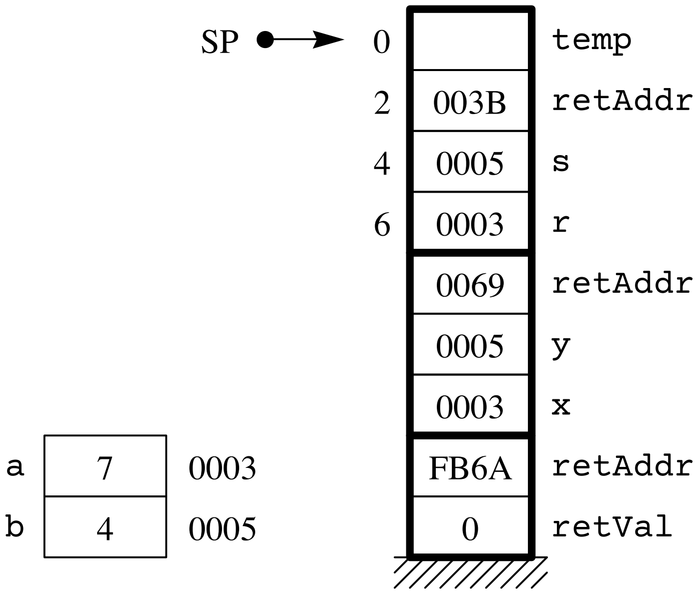


(b) After swap(x, y).

Computer Systems Sixth edition Figure 6.35

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

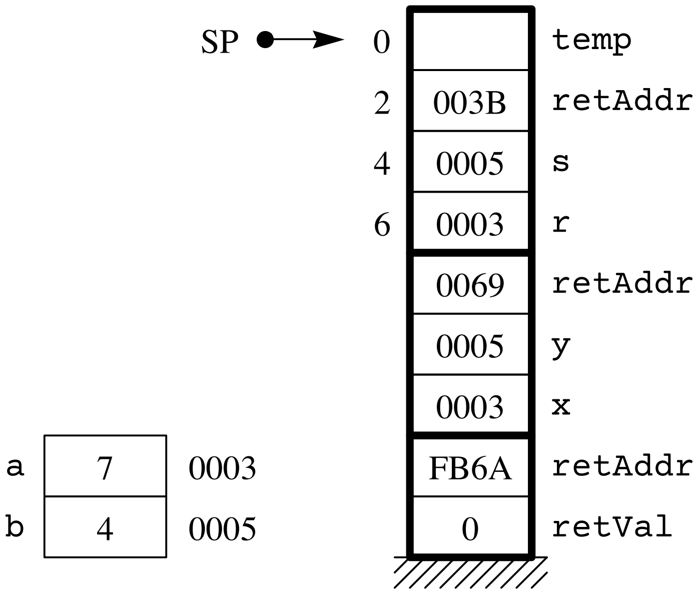
```
;******* void order(int *x, int *y)
x:       .EQUATE 4           ;formal parameter #2h
y:       .EQUATE 2           ;formal parameter #2h
order:   LDWA    x,sf        ;if (*x > *y)
         CPWA    y,sf
         BRLE    endIf
         LDWA    x,s         ;move x
         STWA    -2,s
         LDWA    y,s         ;move y
```



(b) After swap(x, y).

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

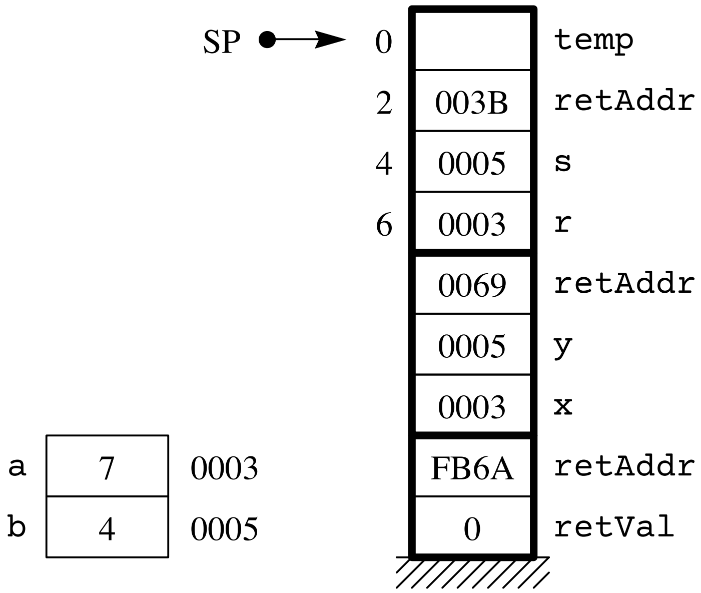
```
;******* void order(int *x, int *y)
x:         .EQUATE 4              ;formal parameter #2h
y:         .EQUATE 2              ;formal parameter #2h
order:     LDWA    x,sf           ;if (*x > *y)
           CPWA    y,sf
           BRLE    endIf
           LDWA    x,s            ;move x
           STWA    -2,s
           LDWA    y,s            ;move y
           STWA    -4,s
```



(b) After `swap(x, y)`.

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

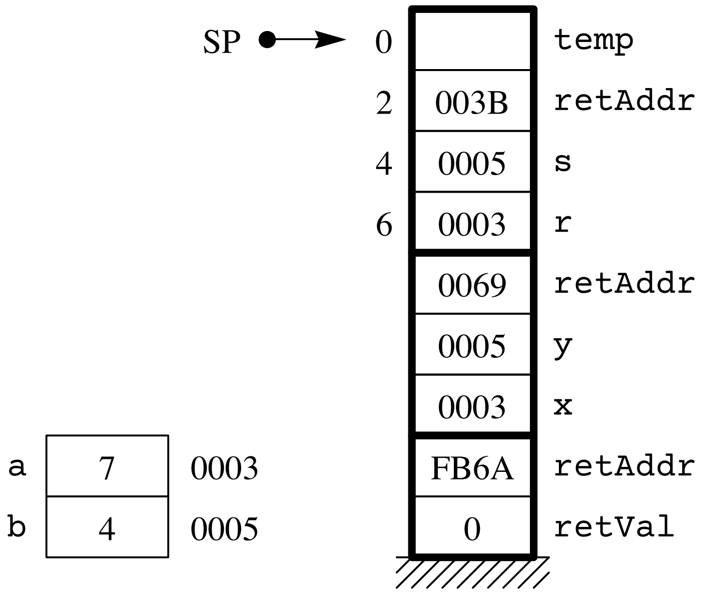
```
;******* void order(int *x, int *y)
x:        .EQUATE 4           ;formal parameter #2h
y:        .EQUATE 2           ;formal parameter #2h
order:    LDWA    x,sf        ;if (*x > *y)
          CPWA    y,sf
          BRLE    endIf
          LDWA    x,s         ;move x
          STWA    -2,s
          LDWA    y,s         ;move y
          STWA    -4,s
          SUBSP   4,i         ;push #r #s
```



(b) After swap(x, y).

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

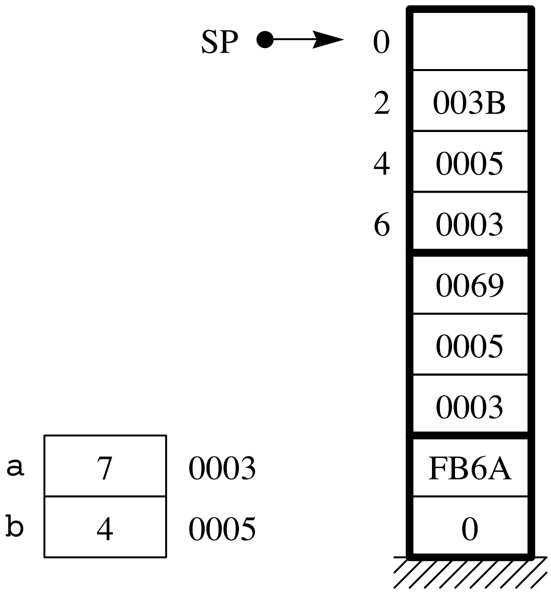
```
;******* void order(int *x, int *y)
x:         .EQUATE 4                ;formal parameter #2h
y:         .EQUATE 2                ;formal parameter #2h
order:     LDWA    x,sf             ;if (*x > *y)
           CPWA    y,sf
           BRLE    endIf
           LDWA    x,s              ;move x
           STWA    -2,s
           LDWA    y,s              ;move y
           STWA    -4,s
           SUBSP   4,i              ;push #r #s
           CALL    swap             ;swap(x, y)
```



(b) After `swap(x, y)`.

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

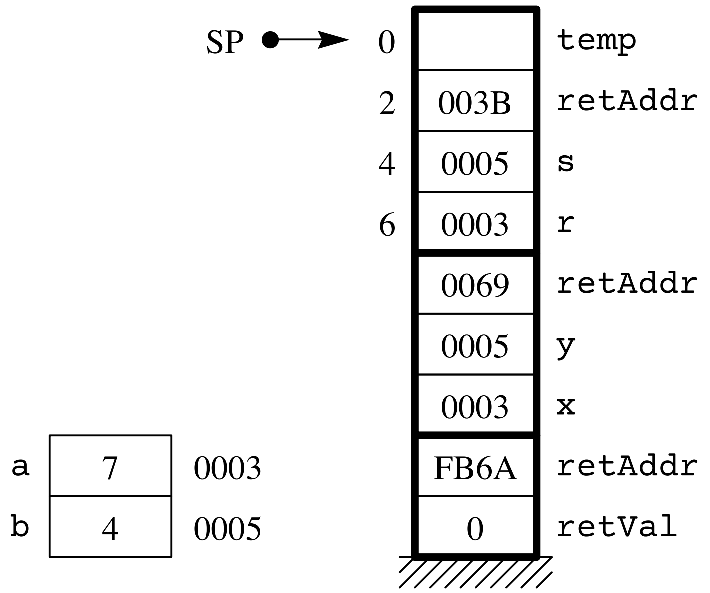
```
;******* void order(int *x, int *y)
x:        .EQUATE 4               ;formal parameter #2h
y:        .EQUATE 2               ;formal parameter #2h
order:    LDWA    x,sf            ;if (*x > *y)
          CPWA    y,sf
          BRLE    endIf
          LDWA    x,s             ;move x
          STWA    -2,s
          LDWA    y,s             ;move y
          STWA    -4,s
          SUBSP   4,i             ;push #r #s
          CALL    swap            ;swap(x, y)
          ADDSP   4,i             ;pop #s #r
```



(b) After swap(x, y).

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

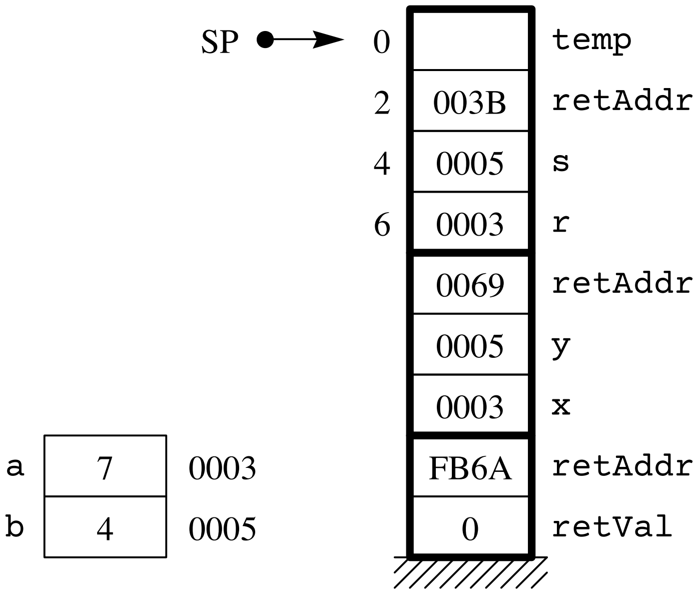
```
;******* void order(int *x, int *y)
x:       .EQUATE 4             ;formal parameter #2h
y:       .EQUATE 2             ;formal parameter #2h
order:   LDWA    x,sf          ;if (*x > *y)
         CPWA    y,sf
         BRLE    endIf
         LDWA    x,s           ;move x
         STWA    -2,s
         LDWA    y,s           ;move y
         STWA    -4,s
         SUBSP   4,i           ;push #r #s
         CALL    swap          ;swap(x, y)
         ADDSP   4,i           ;pop #s #r
endIf:   RET
```



(b) After swap(x, y).

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

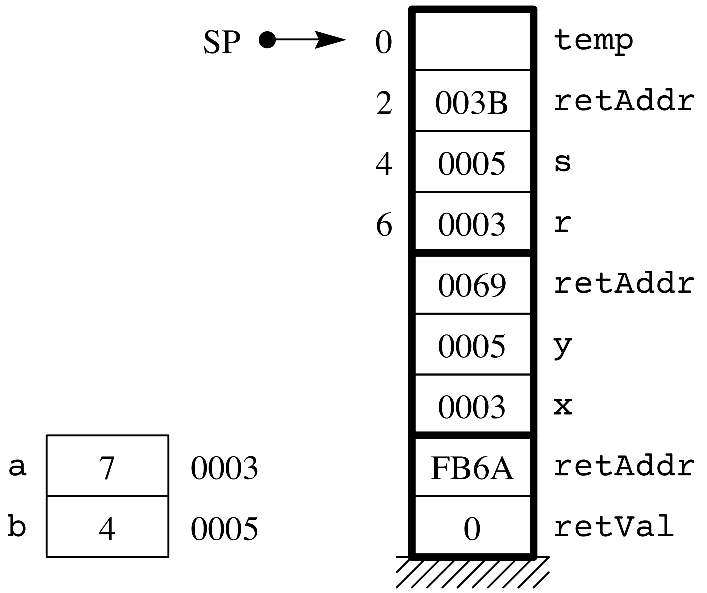
```
;******* void swap(int *r, int *s)
```



(b) After swap(x, y).

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

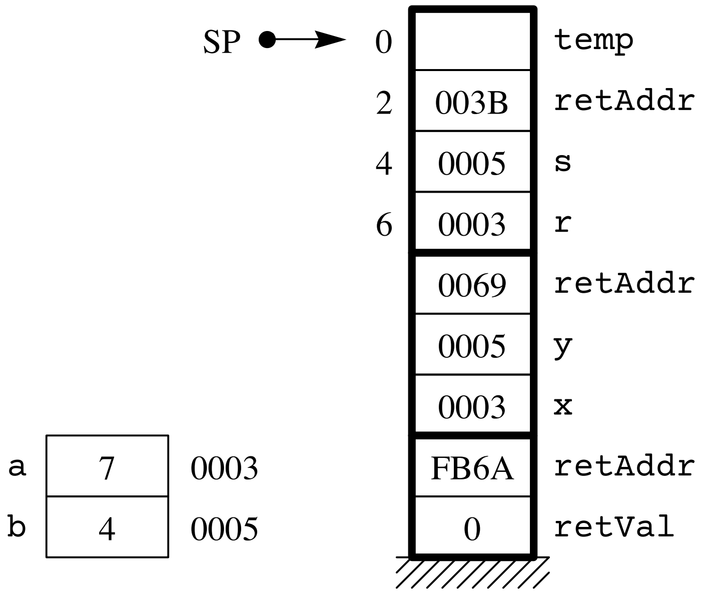
```
;******* void swap(int *r, int *s)
r:         .EQUATE 6              ;formal parameter #2h
```



(b) After `swap(x, y)`.

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

```
;******* void swap(int *r, int *s)
r:          .EQUATE 6           ;formal parameter #2h
s:          .EQUATE 4           ;formal parameter #2h
```



	Value	Name
0		temp
2	003B	retAddr
4	0005	s
6	0003	r
	0069	retAddr
	0005	y
	0003	x
	FB6A	retAddr
	0	retVal

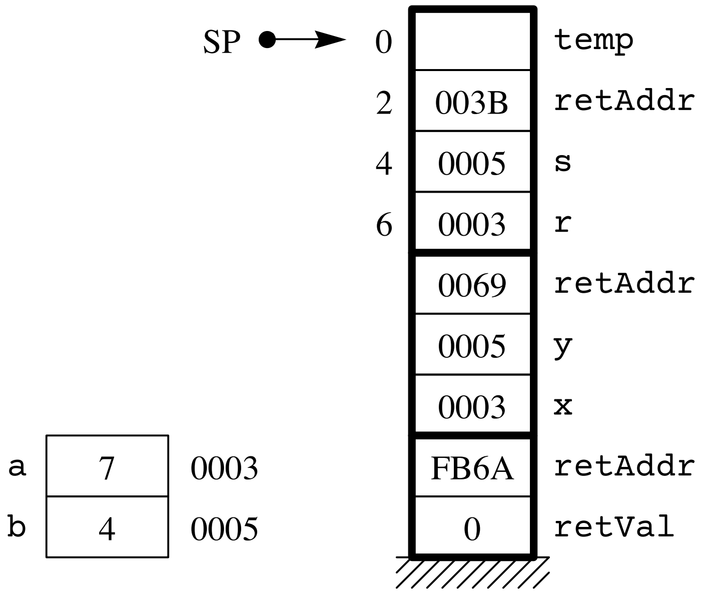
a	7	0003
b	4	0005

(b) After `swap(x, y)`.

Computer Systems Sixth edition Figure 6.35

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

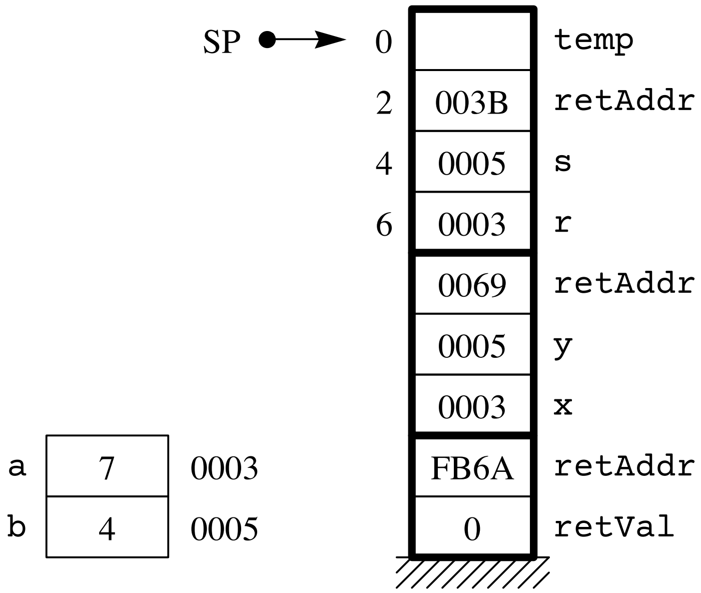
```
;******* void swap(int *r, int *s)
r:         .EQUATE 6              ;formal parameter #2h
s:         .EQUATE 4              ;formal parameter #2h
temp:      .EQUATE 0              ;local variable #2d
```



(b) After `swap(x, y)`.

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

```
;******* void swap(int *r, int *s)
r:        .EQUATE 6           ;formal parameter #2h
s:        .EQUATE 4           ;formal parameter #2h
temp:     .EQUATE 0           ;local variable #2d
swap:     SUBSP   2,i         ;push #temp
```



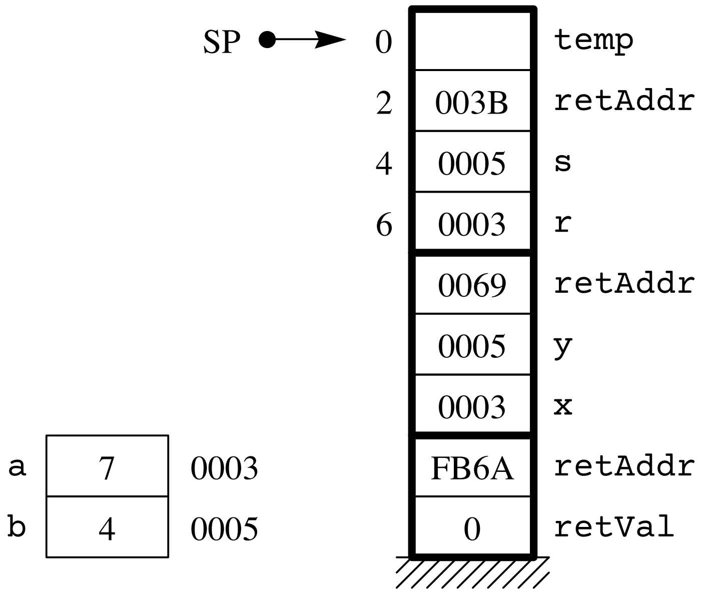
0		temp
2	003B	retAddr
4	0005	s
6	0003	r
	0069	retAddr
	0005	y
	0003	x
	FB6A	retAddr
	0	retVal

a	7	0003
b	4	0005

(b) After `swap(x, y)`.

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

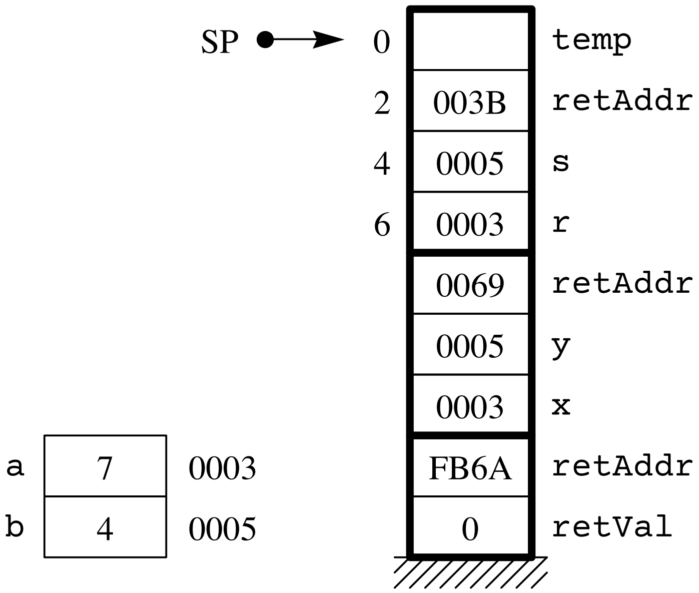
```
;******* void swap(int *r, int *s)
r:         .EQUATE 6          ;formal parameter #2h
s:         .EQUATE 4          ;formal parameter #2h
temp:      .EQUATE 0          ;local variable #2d
swap:      SUBSP   2,i        ;push #temp
           LDWA    r,sf       ;temp = *r
```



(b) After swap(x, y).

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

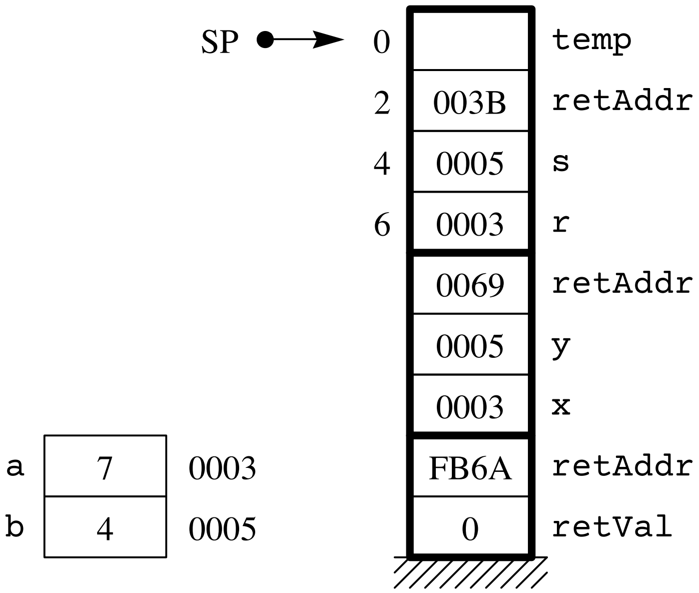
```
;******* void swap(int *r, int *s)
r:       .EQUATE 6          ;formal parameter #2h
s:       .EQUATE 4          ;formal parameter #2h
temp:    .EQUATE 0          ;local variable #2d
swap:    SUBSP   2,i        ;push #temp
         LDWA    r,sf       ;temp = *r
         STWA    temp,s
```



(b) After `swap(x, y)`.

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

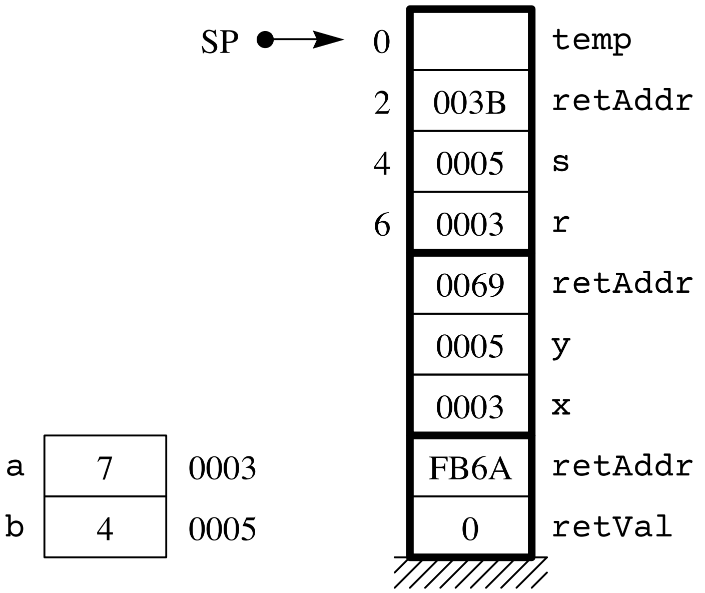
```
;******* void swap(int *r, int *s)
r:         .EQUATE 6             ;formal parameter #2h
s:         .EQUATE 4             ;formal parameter #2h
temp:      .EQUATE 0             ;local variable #2d
swap:      SUBSP   2,i           ;push #temp
           LDWA    r,sf          ;temp = *r
           STWA    temp,s
           LDWA    s,sf          ;*r = *s
```



(b) After swap(x, y).

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

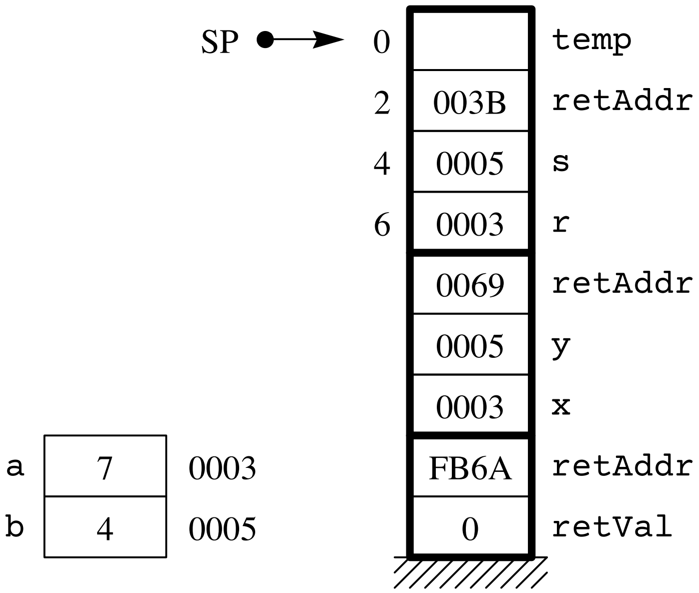
```
;******* void swap(int *r, int *s)
r:       .EQUATE 6          ;formal parameter #2h
s:       .EQUATE 4          ;formal parameter #2h
temp:    .EQUATE 0          ;local variable #2d
swap:    SUBSP   2,i        ;push #temp
         LDWA    r,sf       ;temp = *r
         STWA    temp,s
         LDWA    s,sf       ;*r = *s
         STWA    r,sf
```



(b) After swap(x, y).

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

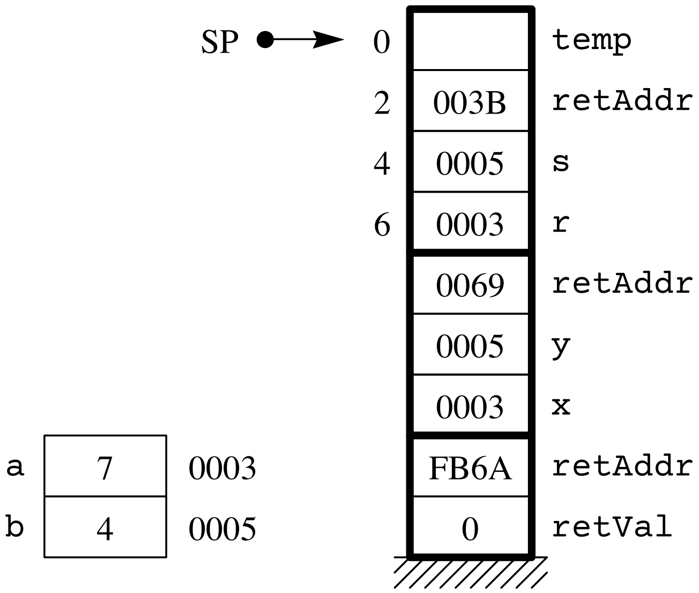
```
;******* void swap(int *r, int *s)
r:       .EQUATE 6          ;formal parameter #2h
s:       .EQUATE 4          ;formal parameter #2h
temp:    .EQUATE 0          ;local variable #2d
swap:    SUBSP   2,i        ;push #temp
         LDWA    r,sf       ;temp = *r
         STWA    temp,s
         LDWA    s,sf       ;*r = *s
         STWA    r,sf
         LDWA    temp,s     ;*s = temp
```



(b) After `swap(x, y)`.

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

```
;******* void swap(int *r, int *s)
r:          .EQUATE 6             ;formal parameter #2h
s:          .EQUATE 4             ;formal parameter #2h
temp:       .EQUATE 0             ;local variable #2d
swap:       SUBSP   2,i           ;push #temp
            LDWA    r,sf          ;temp = *r
            STWA    temp,s
            LDWA    s,sf          ;*r = *s
            STWA    r,sf
            LDWA    temp,s        ;*s = temp
            STWA    s,sf
```

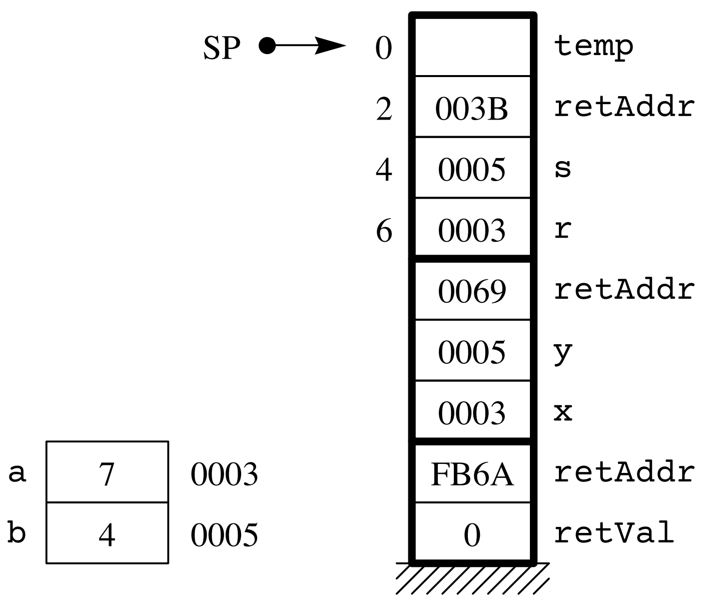


(b) After `swap(x, y)`.

Figure 6.35

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

```
;******* void swap(int *r, int *s)
r:       .EQUATE 6              ;formal parameter #2h
s:       .EQUATE 4              ;formal parameter #2h
temp:    .EQUATE 0              ;local variable #2d
swap:    SUBSP   2,i            ;push #temp
         LDWA    r,sf           ;temp = *r
         STWA    temp,s
         LDWA    s,sf           ;*r = *s
         STWA    r,sf
         LDWA    temp,s         ;*s = temp
         STWA    s,sf
         ADDSP   2,i            ;pop #temp
```



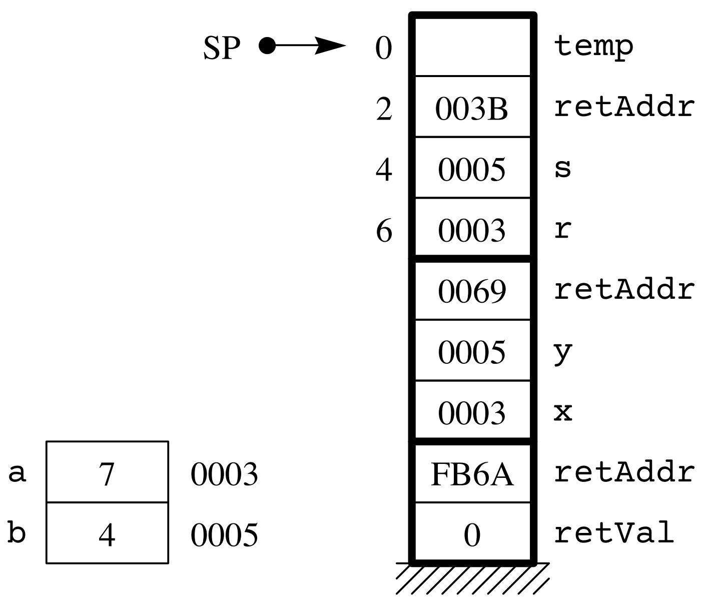
			SP →	0		temp
				2	003B	retAddr
				4	0005	s
				6	0003	r
					0069	retAddr
					0005	y
					0003	x
a	7	0003			FB6A	retAddr
b	4	0005			0	retVal

(b) After swap(x, y).

Figure 6.35

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}
```

```
;******* void swap(int *r, int *s)
r:       .EQUATE 6             ;formal parameter #2h
s:       .EQUATE 4             ;formal parameter #2h
temp:    .EQUATE 0             ;local variable #2d
swap:    SUBSP   2,i           ;push #temp
         LDWA    r,sf          ;temp = *r
         STWA    temp,s
         LDWA    s,sf          ;*r = *s
         STWA    r,sf
         LDWA    temp,s        ;*s = temp
         STWA    s,sf
         ADDSP   2,i           ;pop #temp
         RET
```



(b) After `swap(x, y)`.

The move-SP-to-accumulator instruction

- Instruction specifier: 0000 0101
- Mnemonic: `MOVSPA`
- Accumulator gets the stack pointer

$$\mathrm{A} \leftarrow \mathrm{SP}$$

Translating local pointer parameters

- To access the actual parameter in the caller, generate `MOVSPA` followed by `ADDA` with immediate addressing (`i`)
- To get the formal parameter `x` in the callee, generate load with stack-relative addressing (`s`)
- To get the dereferenced formal parameter `*x` in the callee, generate load with stack-relative deferred addressing (`sf`)

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}

int main() {
    int a, b;
    printf(”Enter an integer: ”);
    scanf(”%d”, &a);
    printf(”Enter an integer: ”);
    scanf(”%d”, &b);
    order(&a, &b);
    printf(”Ordered they are: %d, %d\n”,
        a, b); // ra1
    return 0;
}
```

```
void swap(int *r, int *s) {
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}

void order(int *x, int *y) {
    if (*x > *y) {
        swap(x, y);
    } // ra2
}

int main() {
    int a, b;
    printf(”Enter an integer: ”);
    scanf(”%d”, &a);
    printf(”Enter an integer: ”);
    scanf(”%d”, &b);
    order(&a, &b);
    printf(”Ordered they are: %d, %d\n”,
        a, b); // ra1
    return 0;
}
```

```
         BR      main
;
;******* void swap(int *r, int *s)
```

Identical to `swap( )` *in Figure 6.35.*

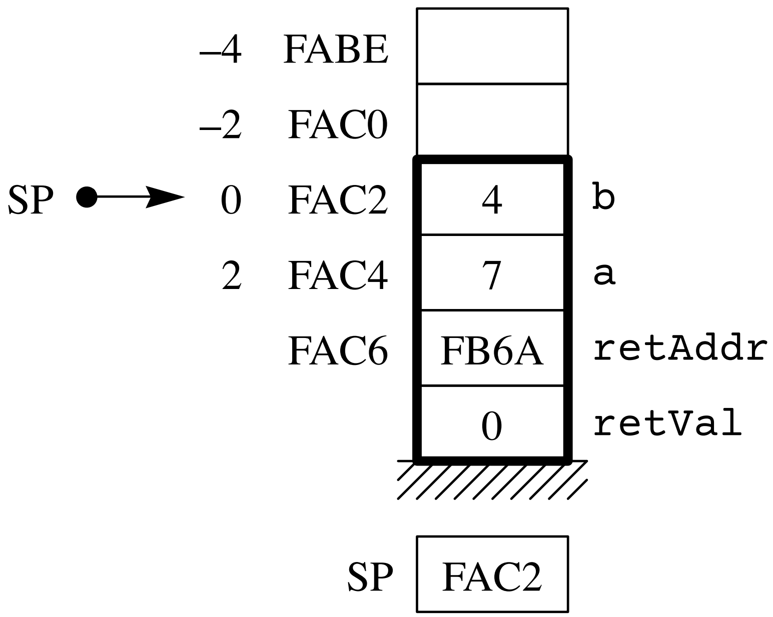
```
;******* void order(int *x, int *y)
```

Identical to `order( )` *in Figure 6.35.*

Figure 6.37

```
int main() {
    int a, b;
    printf(″Enter an integer: ″);
    scanf(″%d″, &a);
    printf(″Enter an integer: ″);
    scanf(″%d″, &b);
    order(&a, &b);
    printf(″Ordered they are: %d, %d\n″,
        a, b); // ra1
    return 0;
}
```

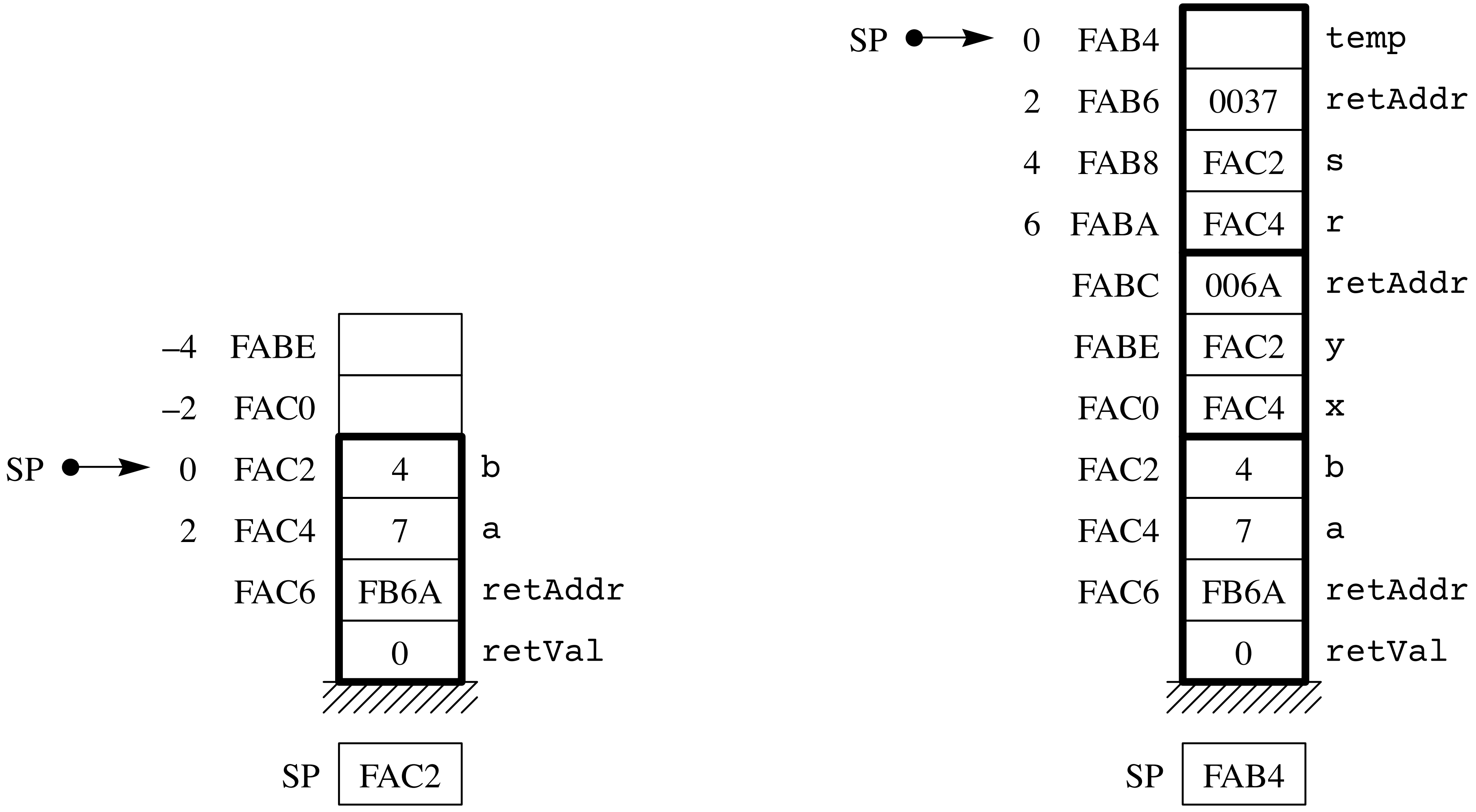
```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```



(a) Before `order(&a, &b)`.

Figure 6.38

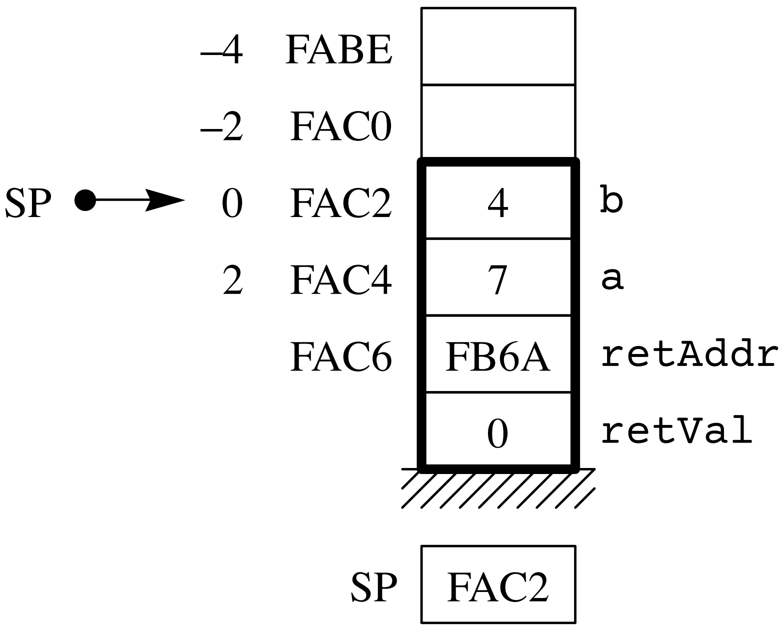
```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```



(a) Before `order(&a, &b)`.

(b) After `swap(x, y)`.

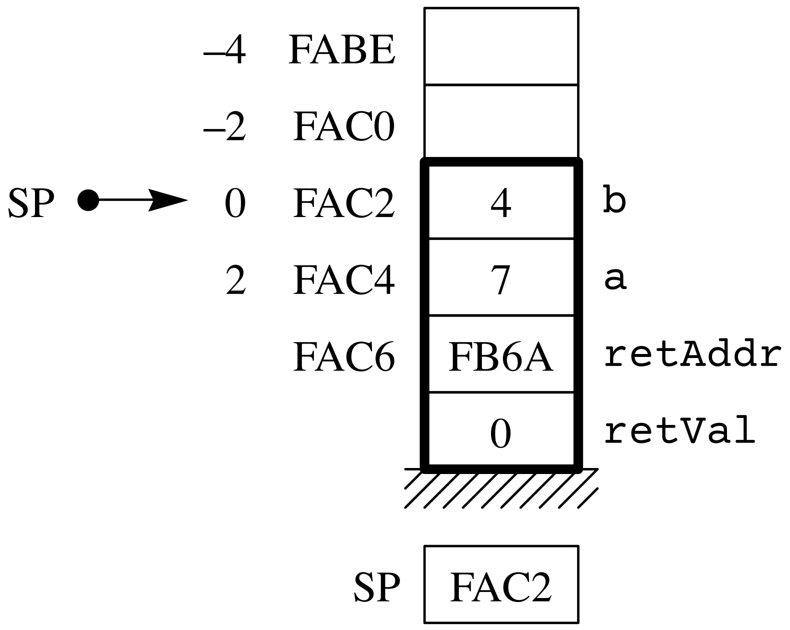
```
int main() {                                  ;******* main()
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

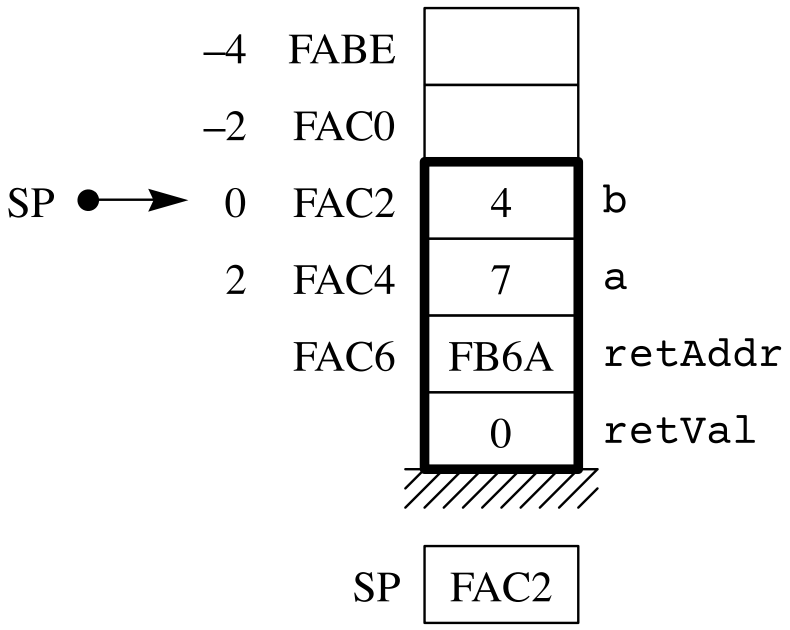
```
;******* main()
a:        .EQUATE 2           ;local variable #2d
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

```
;******* main()
a:       .EQUATE 2          ;local variable #2d
b:       .EQUATE 0          ;local variable #2d
```

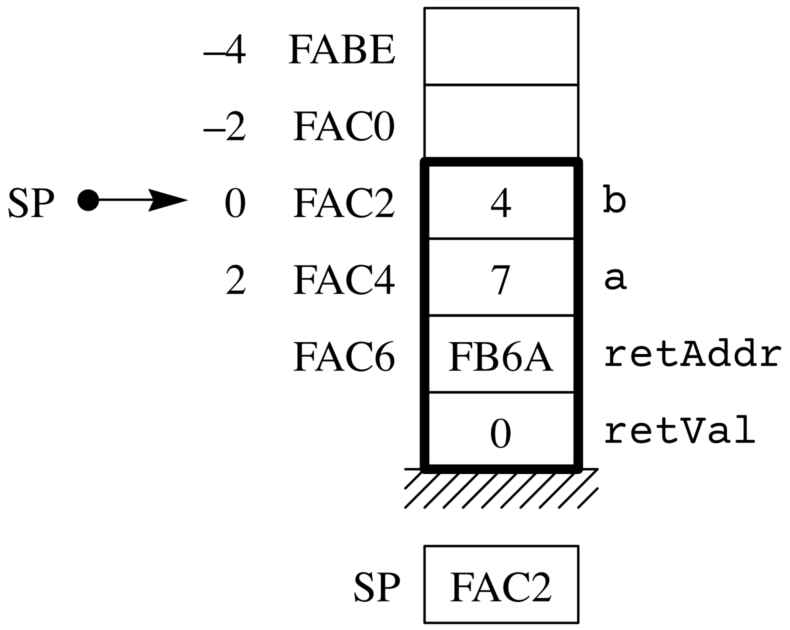


(a) Before `order(&a, &b)`.

Computer Systems Sixth edition Figure 6.37

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

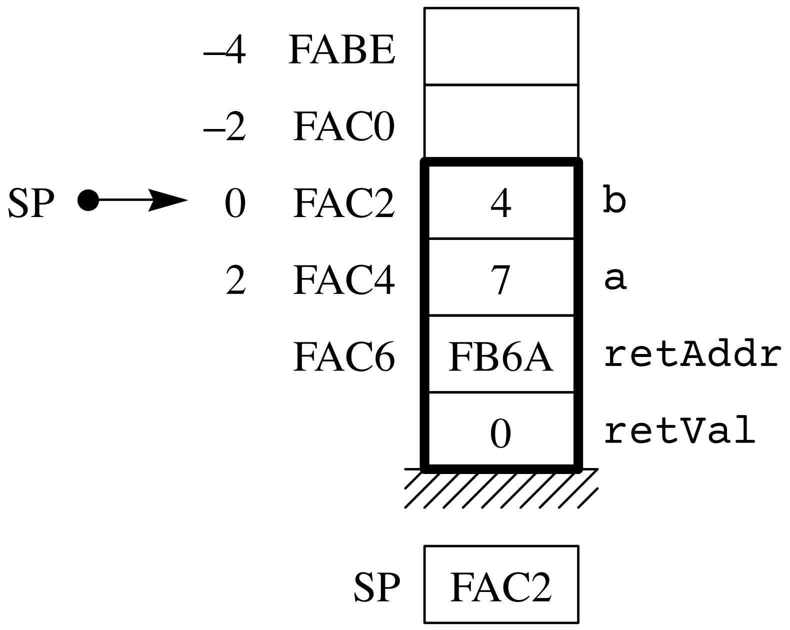
```
;******* main()
a:       .EQUATE 2             ;local variable #2d
b:       .EQUATE 0             ;local variable #2d
main:    SUBSP   4,i           ;push #a #b
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

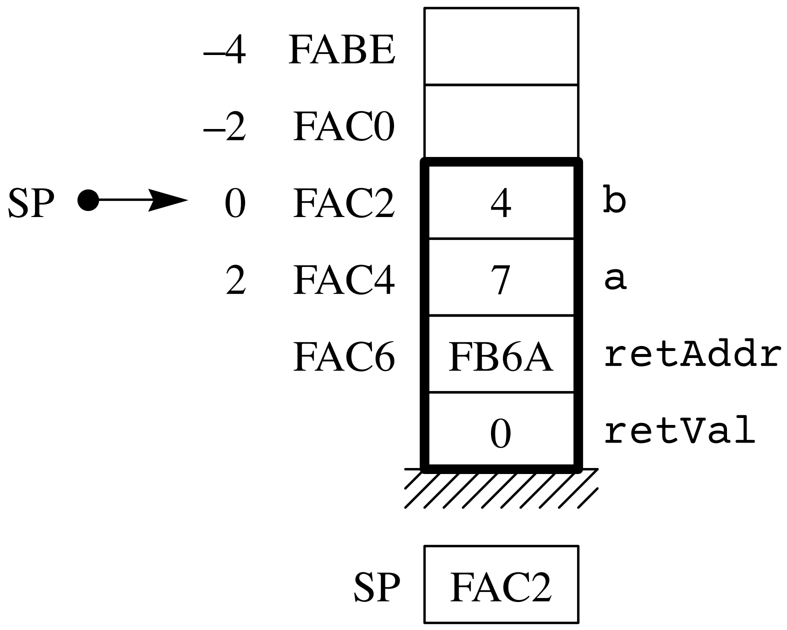
```
;******* main()
a:       .EQUATE 2          ;local variable #2d
b:       .EQUATE 0          ;local variable #2d
main:    SUBSP   4,i        ;push #a #b
         @STRO   msg1,d     ;printf("Enter an ir
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

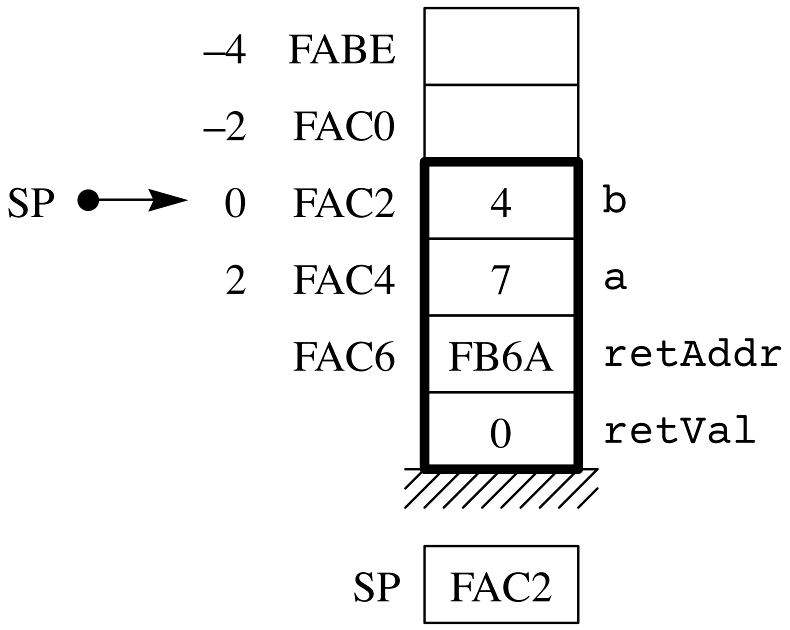
```
;******* main()
a:       .EQUATE 2        ;local variable #2d
b:       .EQUATE 0        ;local variable #2d
main:    SUBSP   4,i      ;push #a #b
         @STRO   msg1,d   ;printf("Enter an ir
         @DECI   a,s      ;scanf("%d", &a)
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

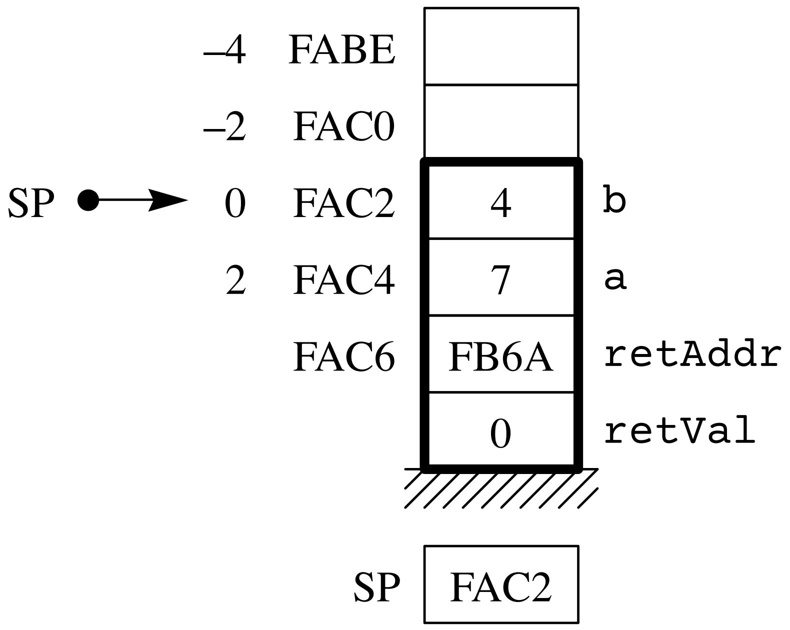
```
;******* main()
a:       .EQUATE 2        ;local variable #2d
b:       .EQUATE 0        ;local variable #2d
main:    SUBSP   4,i      ;push #a #b
         @STRO   msg1,d   ;printf("Enter an in
         @DECI   a,s      ;scanf("%d", &a)
         @STRO   msg1,d   ;printf("Enter an in
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

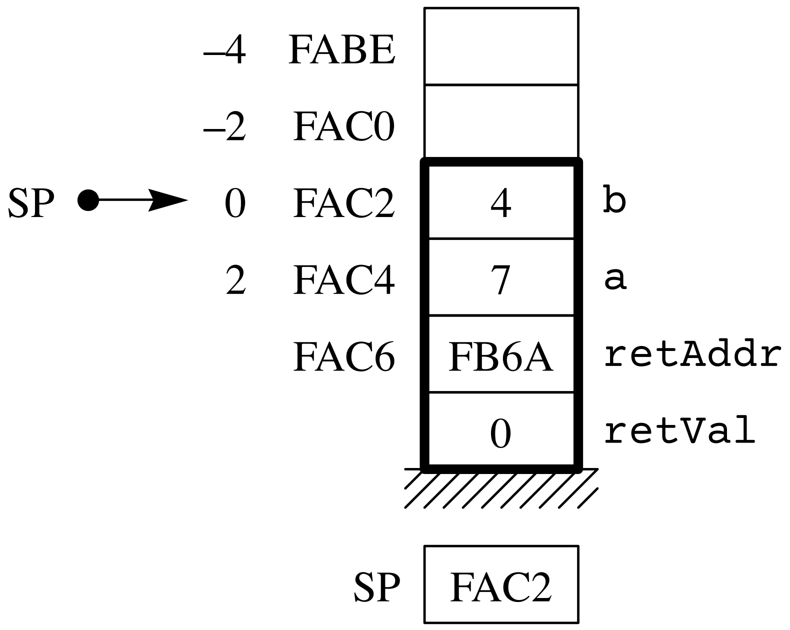
```
;******* main()
a:       .EQUATE 2          ;local variable #2d
b:       .EQUATE 0          ;local variable #2d
main:    SUBSP   4,i        ;push #a #b
         @STRO   msg1,d     ;printf("Enter an in
         @DECI   a,s        ;scanf("%d", &a)
         @STRO   msg1,d     ;printf("Enter an in
         @DECI   b,s        ;scanf("%d", &b)
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

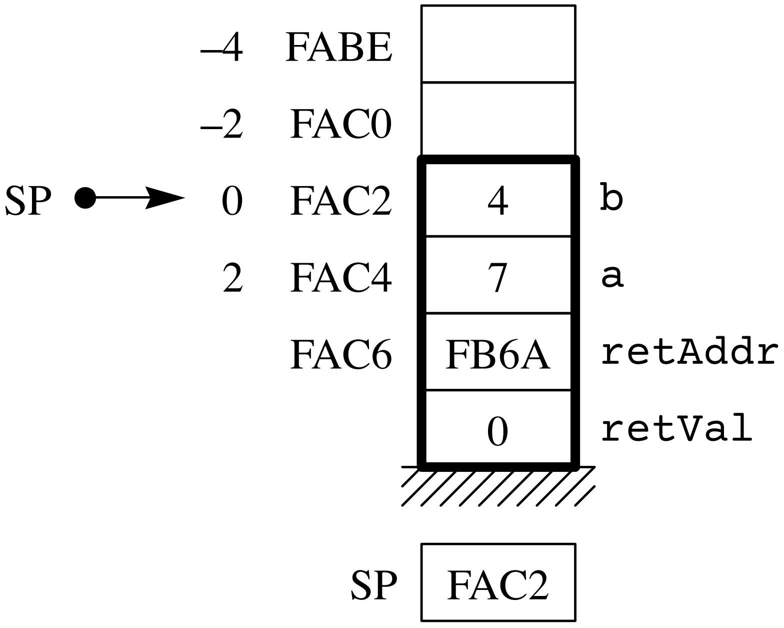
```
;******* main()
a:       .EQUATE 2           ;local variable #2d
b:       .EQUATE 0           ;local variable #2d
main:    SUBSP   4,i         ;push #a #b
         @STRO   msg1,d      ;printf("Enter an in
         @DECI   a,s         ;scanf("%d", &a)
         @STRO   msg1,d      ;printf("Enter an in
         @DECI   b,s         ;scanf("%d", &b)
         MOVSPA              ;move &a
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

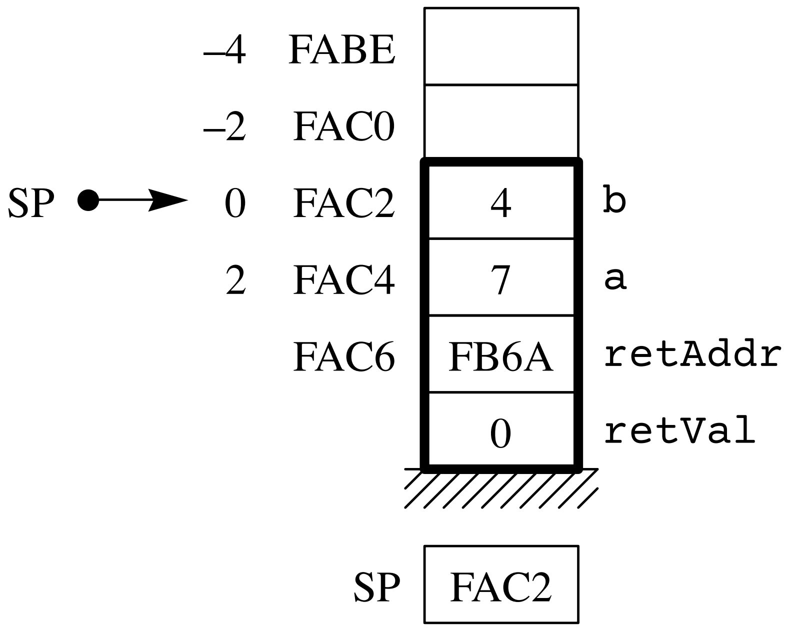
```
;******* main()
a:       .EQUATE 2           ;local variable #2d
b:       .EQUATE 0           ;local variable #2d
main:    SUBSP   4,i         ;push #a #b
         @STRO   msg1,d      ;printf("Enter an in
         @DECI   a,s         ;scanf("%d", &a)
         @STRO   msg1,d      ;printf("Enter an in
         @DECI   b,s         ;scanf("%d", &b)
         MOVSPA              ;move &a
         ADDA    a,i
```



(a) Before order(&a, &b).

```
int main() {
    int a, b;
    printf(”Enter an integer: ”);
    scanf(”%d”, &a);
    printf(”Enter an integer: ”);
    scanf(”%d”, &b);
    order(&a, &b);
    printf(”Ordered they are: %d, %d\n”,
        a, b); // ra1
    return 0;
}
```

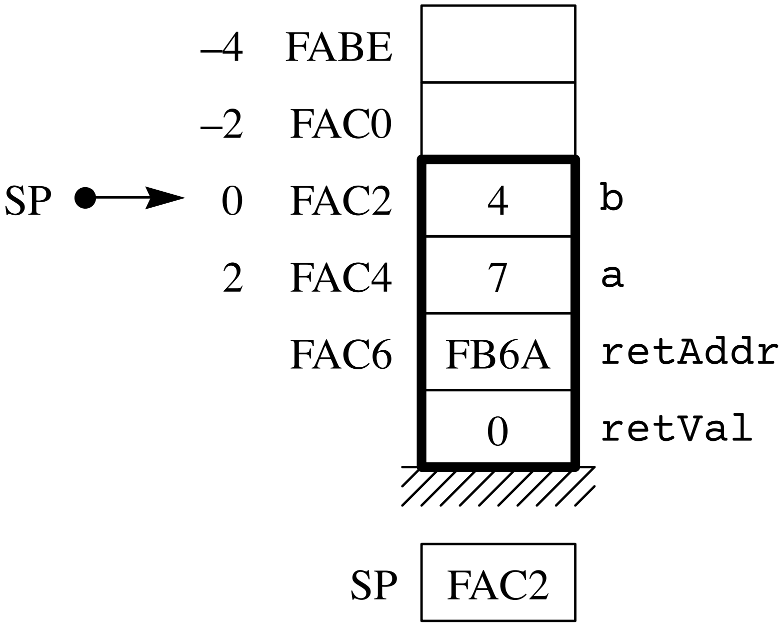
```
;******* main()
a:       .EQUATE 2        ;local variable #2d
b:       .EQUATE 0        ;local variable #2d
main:    SUBSP   4,i      ;push #a #b
         @STRO   msg1,d   ;printf(”Enter an in
         @DECI   a,s      ;scanf(”%d”, &a)
         @STRO   msg1,d   ;printf(”Enter an in
         @DECI   b,s      ;scanf(”%d”, &b)
         MOVSPA           ;move &a
         ADDA    a,i
         STWA    -2,s
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

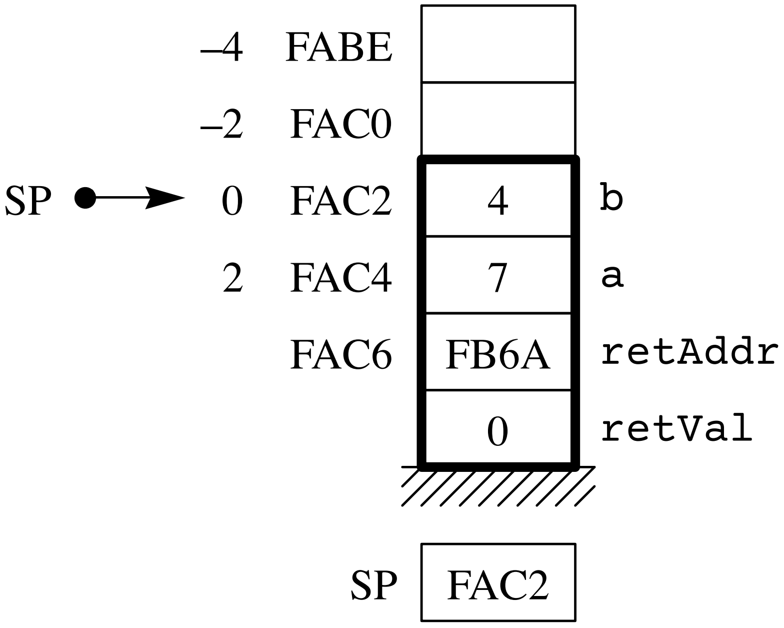
```
;******* main()
a:       .EQUATE 2          ;local variable #2d
b:       .EQUATE 0          ;local variable #2d
main:    SUBSP   4,i        ;push #a #b
         @STRO   msg1,d     ;printf("Enter an in
         @DECI   a,s        ;scanf("%d", &a)
         @STRO   msg1,d     ;printf("Enter an in
         @DECI   b,s        ;scanf("%d", &b)
         MOVSPA             ;move &a
         ADDA    a,i
         STWA    -2,s
         MOVSPA             ;move &b
```



(a) Before order(&a, &b).

```
int main() {
    int a, b;
    printf(”Enter an integer: ”);
    scanf(”%d”, &a);
    printf(”Enter an integer: ”);
    scanf(”%d”, &b);
    order(&a, &b);
    printf(”Ordered they are: %d, %d\n”,
        a, b); // ra1
    return 0;
}
```

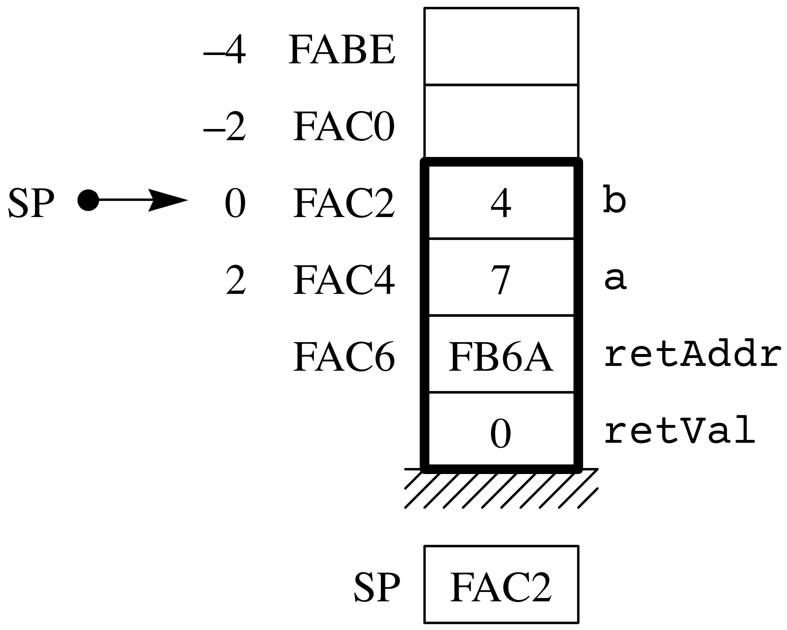
```
;******* main()
a:       .EQUATE 2           ;local variable #2d
b:       .EQUATE 0           ;local variable #2d
main:    SUBSP   4,i         ;push #a #b
         @STRO   msg1,d      ;printf(”Enter an in
         @DECI   a,s         ;scanf(”%d”, &a)
         @STRO   msg1,d      ;printf(”Enter an in
         @DECI   b,s         ;scanf(”%d”, &b)
         MOVSPA              ;move &a
         ADDA    a,i
         STWA    -2,s
         MOVSPA              ;move &b
         ADDA    b,i
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

```
;******* main()
a:       .EQUATE 2          ;local variable #2d
b:       .EQUATE 0          ;local variable #2d
main:    SUBSP   4,i        ;push #a #b
         @STRO   msg1,d     ;printf("Enter an in
         @DECI   a,s        ;scanf("%d", &a)
         @STRO   msg1,d     ;printf("Enter an in
         @DECI   b,s        ;scanf("%d", &b)
         MOVSPA             ;move &a
         ADDA    a,i
         STWA    -2,s
         MOVSPA             ;move &b
         ADDA    b,i
         STWA    -4,s
```

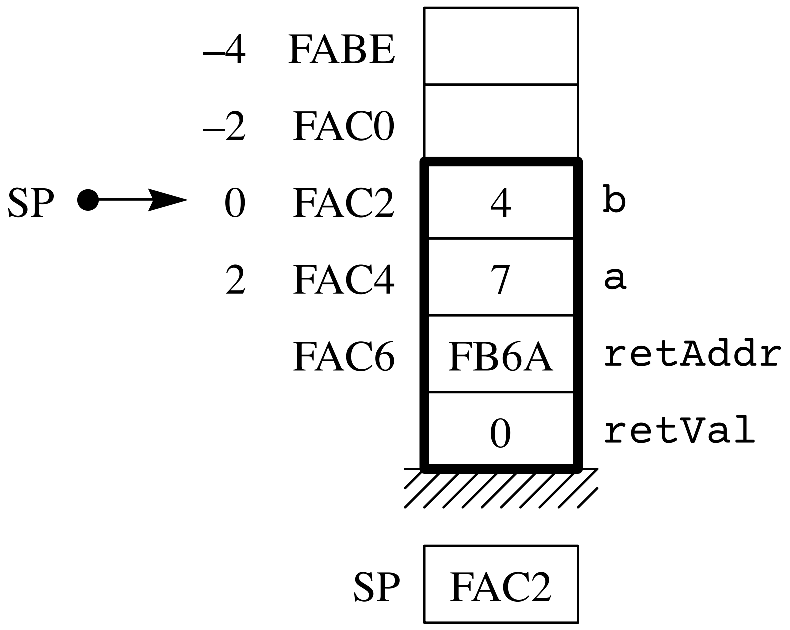


(a) Before `order(&a, &b)`.

Computer Systems Sixth edition Figure 6.37

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

```
;******* main()
a:       .EQUATE 2           ;local variable #2d
b:       .EQUATE 0           ;local variable #2d
main:    SUBSP   4,i         ;push #a #b
         @STRO   msg1,d      ;printf("Enter an ir
         @DECI   a,s         ;scanf("%d", &a)
         @STRO   msg1,d      ;printf("Enter an ir
         @DECI   b,s         ;scanf("%d", &b)
         MOVSPA              ;move &a
         ADDA    a,i
         STWA    -2,s
         MOVSPA              ;move &b
         ADDA    b,i
         STWA    -4,s
         SUBSP   4,i         ;push #x #y
```



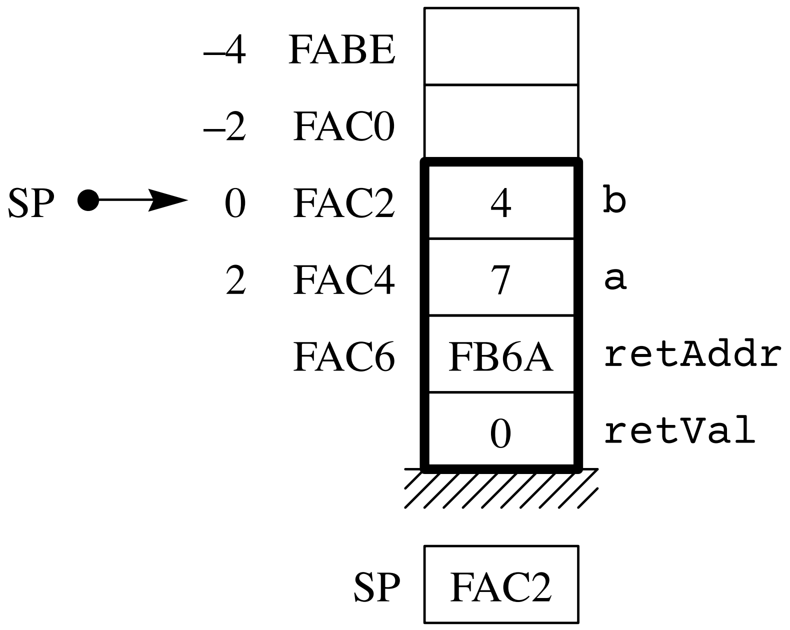
(a) Before `order(&a, &b)`.

Computer Systems Sixth edition

Figure 6.37

```
int main() {
    int a, b;
    printf(”Enter an integer: ”);
    scanf(”%d”, &a);
    printf(”Enter an integer: ”);
    scanf(”%d”, &b);
    order(&a, &b);
    printf(”Ordered they are: %d, %d\n”,
        a, b); // ra1
    return 0;
}
```

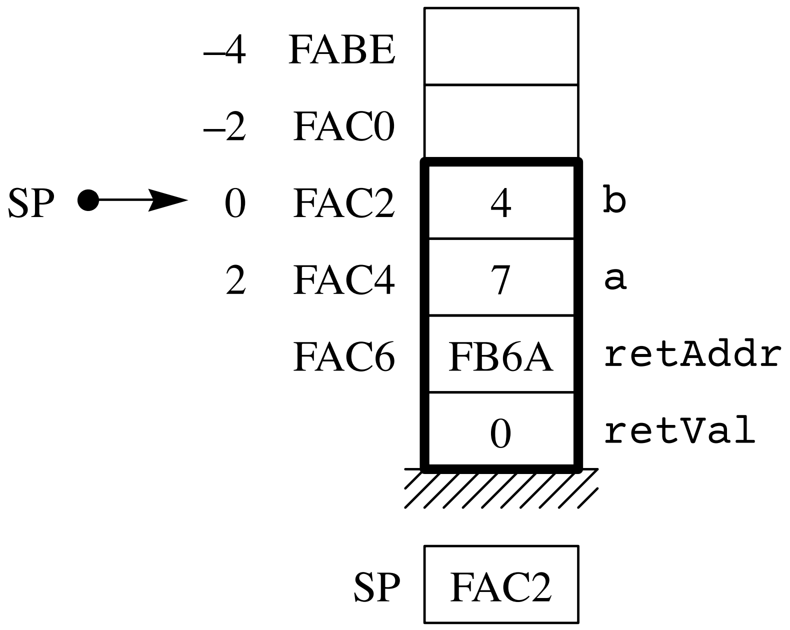
```
;******* main()
a:       .EQUATE 2          ;local variable #2d
b:       .EQUATE 0          ;local variable #2d
main:    SUBSP   4,i        ;push #a #b
         @STRO   msg1,d     ;printf(”Enter an in
         @DECI   a,s        ;scanf(”%d”, &a)
         @STRO   msg1,d     ;printf(”Enter an in
         @DECI   b,s        ;scanf(”%d”, &b)
         MOVSPA             ;move &a
         ADDA    a,i
         STWA    -2,s
         MOVSPA             ;move &b
         ADDA    b,i
         STWA    -4,s
         SUBSP   4,i        ;push #x #y
         CALL    order      ;order(&a, &b)
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf(”Enter an integer: ”);
    scanf(”%d”, &a);
    printf(”Enter an integer: ”);
    scanf(”%d”, &b);
    order(&a, &b);
    printf(”Ordered they are: %d, %d\n”,
        a, b); // ra1
    return 0;
}
```

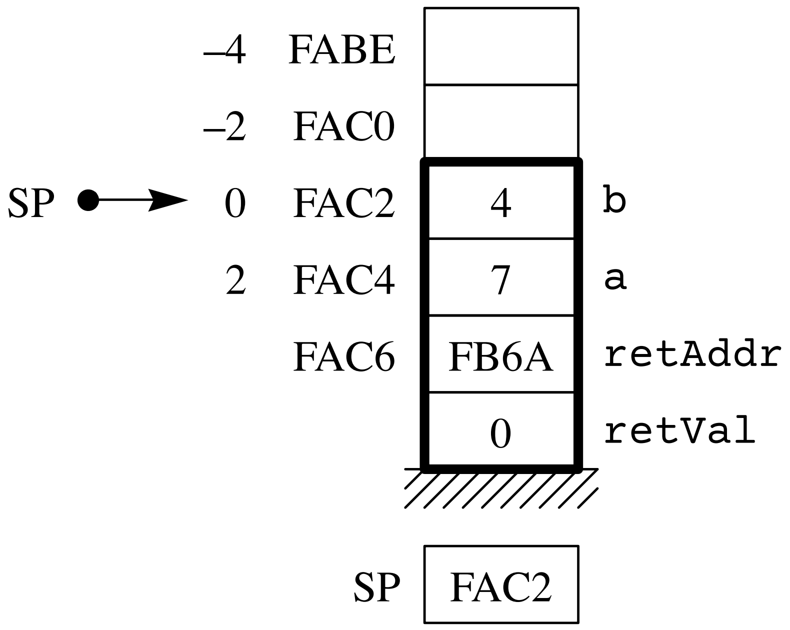
```
;******* main()
a:       .EQUATE 2          ;local variable #2d
b:       .EQUATE 0          ;local variable #2d
main:    SUBSP   4,i        ;push #a #b
         @STRO   msg1,d     ;printf(”Enter an i
         @DECI   a,s        ;scanf(”%d”, &a)
         @STRO   msg1,d     ;printf(”Enter an i
         @DECI   b,s        ;scanf(”%d”, &b)
         MOVSPA             ;move &a
         ADDA    a,i
         STWA    -2,s
         MOVSPA             ;move &b
         ADDA    b,i
         STWA    -4,s
         SUBSP   4,i        ;push #x #y
         CALL    order      ;order(&a, &b)
ra1:     ADDSP   4,i        ;pop #y #x
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

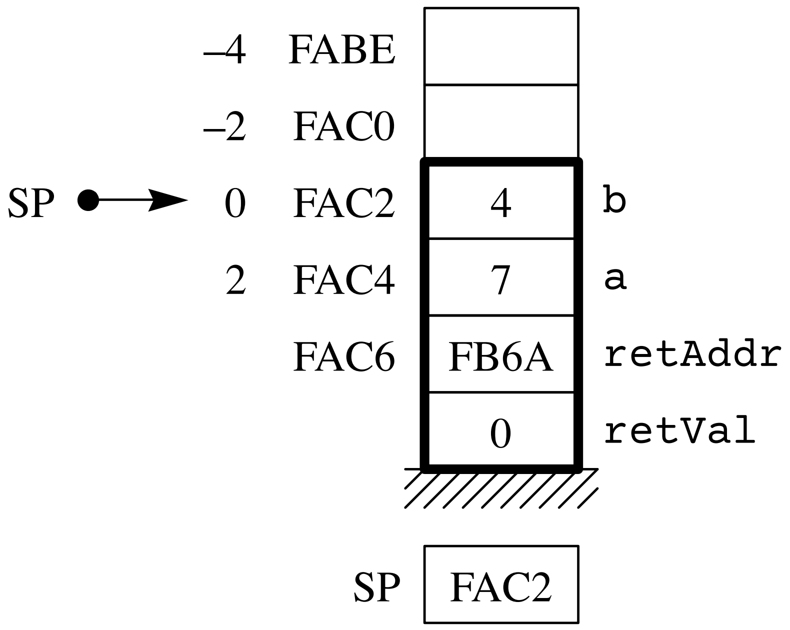
```
;******* main()
a:       .EQUATE 2          ;local variable #2d
b:       .EQUATE 0          ;local variable #2d
main:    SUBSP   4,i        ;push #a #b
         @STRO   msg1,d     ;printf("Enter an i
         @DECI   a,s        ;scanf("%d", &a)
         @STRO   msg1,d     ;printf("Enter an i
         @DECI   b,s        ;scanf("%d", &b)
         MOVSPA             ;move &a
         ADDA    a,i
         STWA    -2,s
         MOVSPA             ;move &b
         ADDA    b,i
         STWA    -4,s
         SUBSP   4,i        ;push #x #y
         CALL    order      ;order(&a, &b)
ra1:     ADDSP   4,i        ;pop #y #x
         @STRO   msg2,d     ;printf("Ordered the
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

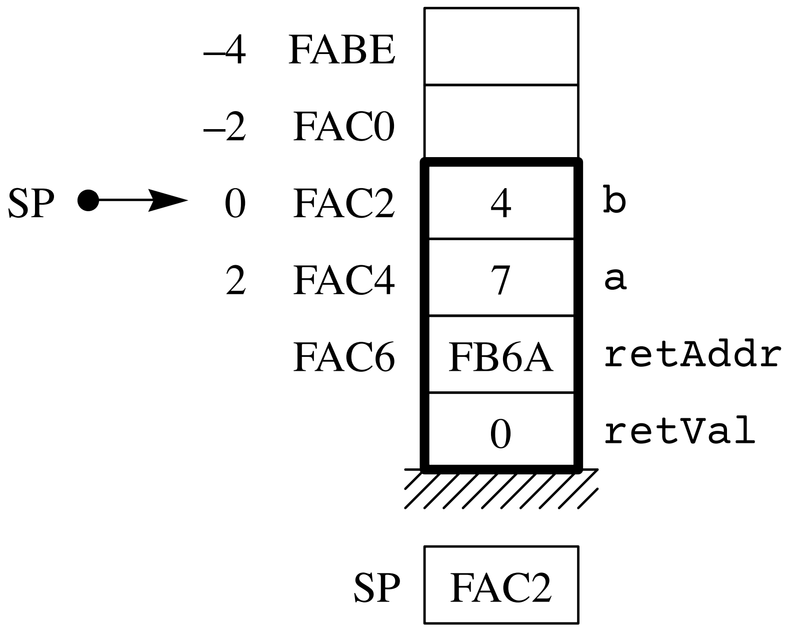
```
;******* main()
a:       .EQUATE 2           ;local variable #2d
b:       .EQUATE 0           ;local variable #2d
main:    SUBSP   4,i         ;push #a #b
         @STRO   msg1,d      ;printf("Enter an ir
         @DECI   a,s         ;scanf("%d", &a)
         @STRO   msg1,d      ;printf("Enter an ir
         @DECI   b,s         ;scanf("%d", &b)
         MOVSPA              ;move &a
         ADDA    a,i
         STWA    -2,s
         MOVSPA              ;move &b
         ADDA    b,i
         STWA    -4,s
         SUBSP   4,i         ;push #x #y
         CALL    order       ;order(&a, &b)
ra1:     ADDSP   4,i         ;pop #y #x
         @STRO   msg2,d      ;printf("Ordered the
         @DECO   a,s         ;, a
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

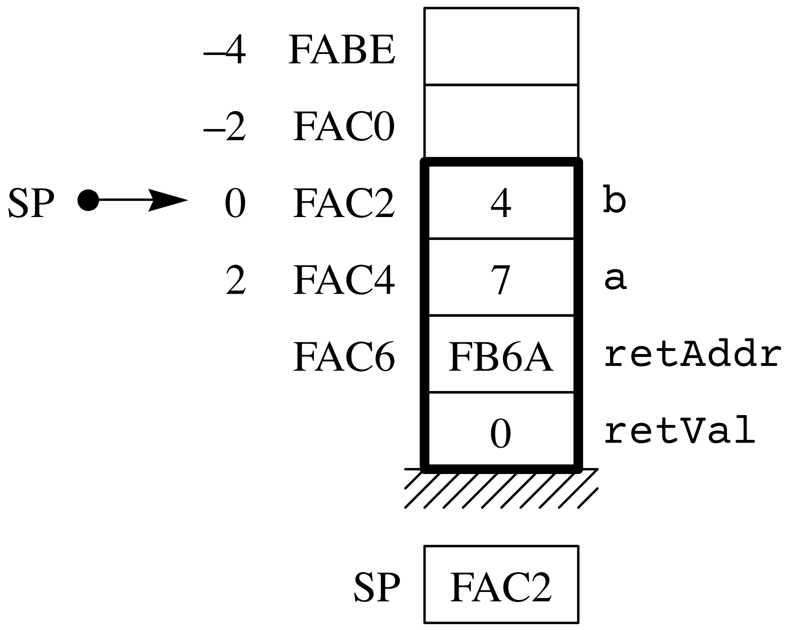
```
;******* main()
a:       .EQUATE 2          ;local variable #2d
b:       .EQUATE 0          ;local variable #2d
main:    SUBSP   4,i        ;push #a #b
         @STRO   msg1,d     ;printf("Enter an in
         @DECI   a,s        ;scanf("%d", &a)
         @STRO   msg1,d     ;printf("Enter an in
         @DECI   b,s        ;scanf("%d", &b)
         MOVSPA             ;move &a
         ADDA    a,i
         STWA    -2,s
         MOVSPA             ;move &b
         ADDA    b,i
         STWA    -4,s
         SUBSP   4,i        ;push #x #y
         CALL    order      ;order(&a, &b)
ra1:     ADDSP   4,i        ;pop #y #x
         @STRO   msg2,d     ;printf("Ordered the
         @DECO   a,s        ;, a
         @STRO   msg3,d
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

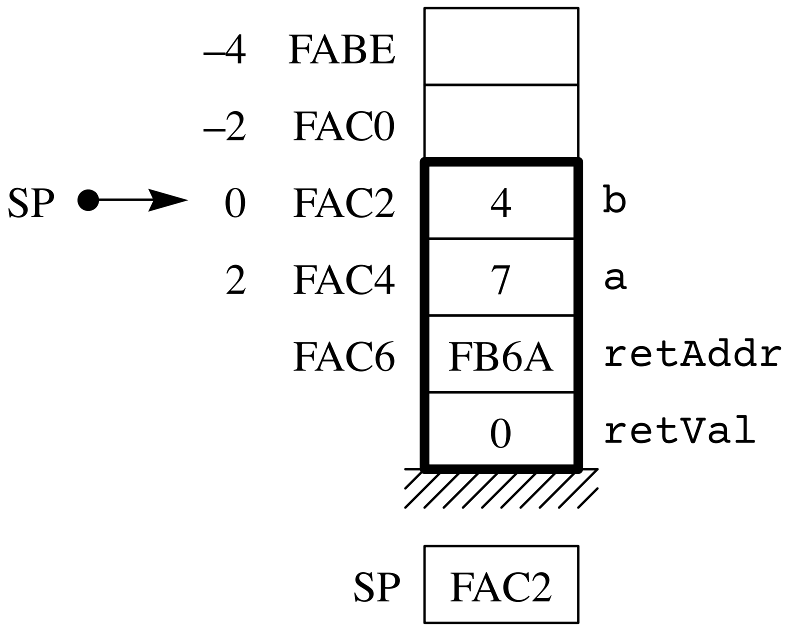
```
;******* main()
a:       .EQUATE 2          ;local variable #2d
b:       .EQUATE 0          ;local variable #2d
main:    SUBSP   4,i        ;push #a #b
         @STRO   msg1,d     ;printf("Enter an in
         @DECI   a,s        ;scanf("%d", &a)
         @STRO   msg1,d     ;printf("Enter an in
         @DECI   b,s        ;scanf("%d", &b)
         MOVSPA             ;move &a
         ADDA    a,i
         STWA    -2,s
         MOVSPA             ;move &b
         ADDA    b,i
         STWA    -4,s
         SUBSP   4,i        ;push #x #y
         CALL    order      ;order(&a, &b)
ra1:     ADDSP   4,i        ;pop #y #x
         @STRO   msg2,d     ;printf("Ordered the
         @DECO   a,s        ;, a
         @STRO   msg3,d
         @DECO   b,s        ;, b)
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf(”Enter an integer: ”);
    scanf(”%d”, &a);
    printf(”Enter an integer: ”);
    scanf(”%d”, &b);
    order(&a, &b);
    printf(”Ordered they are: %d, %d\n”,
        a, b); // ra1
    return 0;
}
```

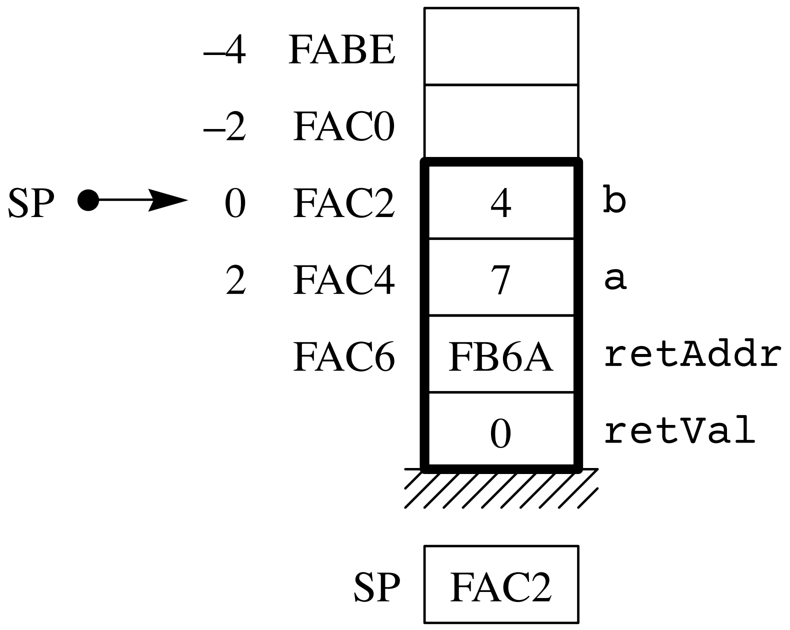
```
;******* main()
a:       .EQUATE 2          ;local variable #2d
b:       .EQUATE 0          ;local variable #2d
main:    SUBSP   4,i        ;push #a #b
         @STRO   msg1,d     ;printf(”Enter an ir
         @DECI   a,s        ;scanf(”%d”, &a)
         @STRO   msg1,d     ;printf(”Enter an ir
         @DECI   b,s        ;scanf(”%d”, &b)
         MOVSPA             ;move &a
         ADDA    a,i
         STWA    -2,s
         MOVSPA             ;move &b
         ADDA    b,i
         STWA    -4,s
         SUBSP   4,i        ;push #x #y
         CALL    order      ;order(&a, &b)
ra1:     ADDSP   4,i        ;pop #y #x
         @STRO   msg2,d     ;printf(”Ordered the
         @DECO   a,s        ;, a
         @STRO   msg3,d
         @DECO   b,s        ;, b)
         @CHARO  ’\n’,i
```



(a) Before `order(&a, &b)`.

```
int main() {
    int a, b;
    printf(”Enter an integer: ”);
    scanf(”%d”, &a);
    printf(”Enter an integer: ”);
    scanf(”%d”, &b);
    order(&a, &b);
    printf(”Ordered they are: %d, %d\n”,
        a, b); // ra1
    return 0;
}
```

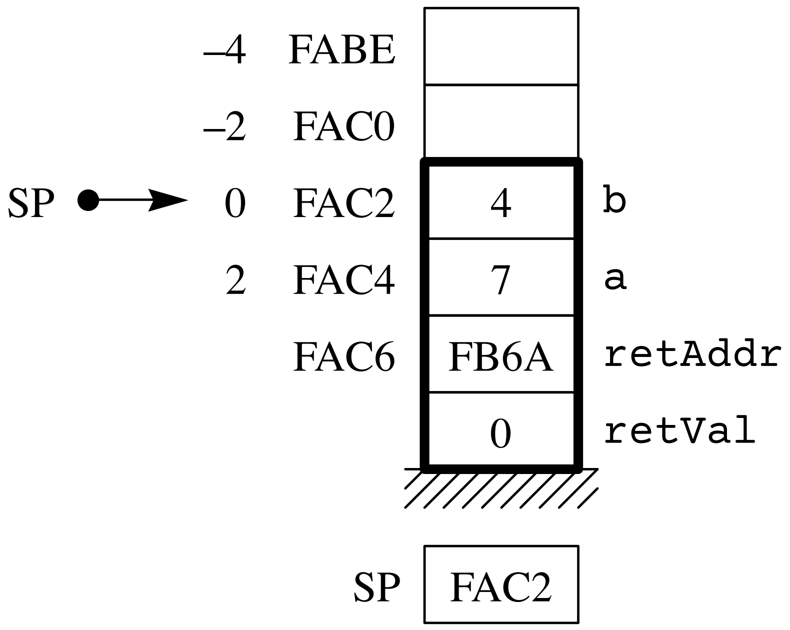
```
;******* main()
a:       .EQUATE 2           ;local variable #2d
b:       .EQUATE 0           ;local variable #2d
main:    SUBSP   4,i         ;push #a #b
         @STRO   msg1,d      ;printf(”Enter an in
         @DECI   a,s         ;scanf(”%d”, &a)
         @STRO   msg1,d      ;printf(”Enter an in
         @DECI   b,s         ;scanf(”%d”, &b)
         MOVSPA              ;move &a
         ADDA    a,i
         STWA    -2,s
         MOVSPA              ;move &b
         ADDA    b,i
         STWA    -4,s
         SUBSP   4,i         ;push #x #y
         CALL    order       ;order(&a, &b)
ra1:     ADDSP   4,i         ;pop #y #x
         @STRO   msg2,d      ;printf(”Ordered the
         @DECO   a,s         ;, a
         @STRO   msg3,d
         @DECO   b,s         ;, b)
         @CHARO  ’\n’,i
         ADDSP   4,i         ;pop #b #a
```



(a) Before order(&a, &b).

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

```
;******* main()
a:       .EQUATE 2          ;local variable #2d
b:       .EQUATE 0          ;local variable #2d
main:    SUBSP   4,i        ;push #a #b
         @STRO   msg1,d     ;printf("Enter an in
         @DECI   a,s        ;scanf("%d", &a)
         @STRO   msg1,d     ;printf("Enter an in
         @DECI   b,s        ;scanf("%d", &b)
         MOVSPA             ;move &a
         ADDA    a,i
         STWA    -2,s
         MOVSPA             ;move &b
         ADDA    b,i
         STWA    -4,s
         SUBSP   4,i        ;push #x #y
         CALL    order      ;order(&a, &b)
ra1:     ADDSP   4,i        ;pop #y #x
         @STRO   msg2,d     ;printf("Ordered the
         @DECO   a,s        ;, a
         @STRO   msg3,d
         @DECO   b,s        ;, b)
         @CHARO  '\n',i
         ADDSP   4,i        ;pop #b #a
         RET
```

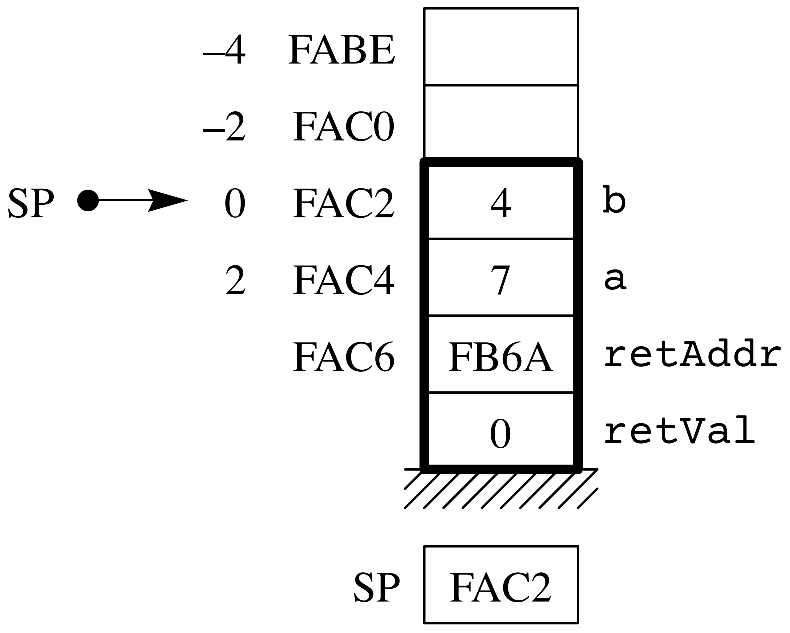


(a) Before `order(&a, &b)`.

Computer Systems Sixth edition Figure 6.37

```
int main() {
    int a, b;
    printf("Enter an integer: ");
    scanf("%d", &a);
    printf("Enter an integer: ");
    scanf("%d", &b);
    order(&a, &b);
    printf("Ordered they are: %d, %d\n",
        a, b); // ra1
    return 0;
}
```

```
;******* main()
a:       .EQUATE 2          ;local variable #2d
b:       .EQUATE 0          ;local variable #2d
main:    SUBSP   4,i        ;push #a #b
         @STRO   msg1,d     ;printf("Enter an in
         @DECI   a,s        ;scanf("%d", &a)
         @STRO   msg1,d     ;printf("Enter an in
         @DECI   b,s        ;scanf("%d", &b)
         MOVSPA             ;move &a
         ADDA    a,i
         STWA    -2,s
         MOVSPA             ;move &b
         ADDA    b,i
         STWA    -4,s
         SUBSP   4,i        ;push #x #y
         CALL    order      ;order(&a, &b)
ra1:     ADDSP   4,i        ;pop #y #x
         @STRO   msg2,d     ;printf("Ordered the
         @DECO   a,s        ;, a
         @STRO   msg3,d
         @DECO   b,s        ;, b)
         @CHARO  '\n',i
         ADDSP   4,i        ;pop #b #a
         RET
msg1:    .ASCII  "Enter an integer: \0"
msg2:    .ASCII  "Ordered they are: \0"
msg3:    .ASCII  ", \0"
```



(a) Before order(&a, &b).