Computer Systems

Sixth edition

Chapter 5

Assembly Language

An Assembler at Level 3

- No operating system
- Bare metal mode
- Used only to introduce assembly language

Two types of bit patterns

- Instructions
  - Mnemonics for opcodes
  - Letters for addressing modes
- Data
  - Pseudo-ops, also called dot commands

Instruction Specifier	Mnemonic	Instruction	Addressing Modes	Status Bits
0000 0000		Illegal instruction		
0000 0001	`RET`	Return from `CALL`	Monadic	
0000 0010	`SRET`	Return from system `CALL`	Monadic	
0000 0011	`MOVFLGA`	Move NZVC flags to A[12 : 15]	Monadic	
0000 0100	`MOVAFLG`	Move A[12 : 15] to NZVC flags	Monadic	NZVC
0000 0101	`MOVSPA`	Move SP to A	Monadic	
0000 0110	`MOVASP`	Move A to SP	Monadic	
0000 0111	`NOP`	No operation	Monadic	
0001 100r	`NEGr`	Negate r	Monadic	NZVC
0001 101r	`ASLr`	Arithmetic shift left r	Monadic	NZVC
0001 110r	`ASRr`	Arithmetic shift right r	Monadic	NZVC
0001 111r	`NOTr`	Bitwise NOT r	Monadic	NZ
0010 000r	`ROLr`	Rotate left r	Monadic	NZC
0010 001r	`RORr`	Rotate right r	Monadic	NZC

Instruction Specifier	Mnemonic	Instruction	Addressing Modes	Status Bits
0010 010a	`BR`	Branch unconditional	i, x	
0010 011a	`BRLE`	Branch if less than or equal to	i, x	
0010 100a	`BRLT`	Branch if less than	i, x	
0010 101a	`BREQ`	Branch if equal to	i, x	
0010 110a	`BRNE`	Branch if not equal to	i, x	
0010 111a	`BRGE`	Branch if greater than or equal to	i, x	
0011 000a	`BRGT`	Branch if greater than	i, x	
0011 001a	`BRV`	Branch if V	i, x	
0011 010a	`BRC`	Branch if C	i, x	
0011 011a	`CALL`	Call subroutine	i, x	
0011 1aaa	`SCALL`	System call	i, d, n, s, sf, x, sx, sfx	
0100 0aaa	`ADDSP`	Add to SP	i, d, n, s, sf, x, sx, sfx	
0100 1aaa	`SUBSP`	Subtract from SP	i, d, n, s, sf, x, sx, sfx	
0101 raaa	`ADDr`	Add to r	i, d, n, s, sf, x, sx, sfx	NZVC
0110 raaa	`SUBr`	Subtract from r	i, d, n, s, sf, x, sx, sfx	NZVC
0111 raaa	`ANDr`	Bitwise AND to r	i, d, n, s, sf, x, sx, sfx	NZ
1000 raaa	`ORr`	Bitwise OR to r	i, d, n, s, sf, x, sx, sfx	NZ
1001 raaa	`XORr`	Bitwise XOR to r	i, d, n, s, sf, x, sx, sfx	NZ
1010 raaa	`CPWr`	Compare word to r	i, d, n, s, sf, x, sx, sfx	NZVC
1011 raaa	`CPBr`	Compare byte to r[8 : 15]	i, d, n, s, sf, x, sx, sfx	NZVC
1100 raaa	`LDWr`	Load word r from memory	i, d, n, s, sf, x, sx, sfx	NZ
1101 raaa	`LDBr`	Load byte r[8 : 15] from memory	i, d, n, s, sf, x, sx, sfx	NZ
1110 raaa	`STWr`	Store word r to memory	d, n, s, sf, x, sx, sfx	
1111 raaa	`STBr`	Store byte r[8 : 15] to memory	d, n, s, sf, x, sx, sfx	

Letters specify the addressing mode

aaa	Addressing Mode	Letters
000	Immediate	`i`
001	Direct	`d`
010	Indirect	`n`
011	Stack-relative	`s`
100	Stack-relative deferred	`sf`
101	Indexed	`x`
110	Stack-indexed	`sx`
111	Stack-deferred indexed	`sfx`

```
1100 0011 0000 0000 1001 1010
1100 0110 0000 0000 1001 1010
1100 1011 0000 0000 1001 1010
1100 1110 0000 0000 1001 1010
```

```
1100 0011 0000 0000 1001 1010    LDWA 0x009A,s
1100 0110 0000 0000 1001 1010
1100 1011 0000 0000 1001 1010
1100 1110 0000 0000 1001 1010
```

```
1100 0011 0000 0000 1001 1010    LDWA 0x009A,s
1100 0110 0000 0000 1001 1010    LDWA 0x009A,sx
1100 1011 0000 0000 1001 1010
1100 1110 0000 0000 1001 1010
```

```
1100 0011 0000 0000 1001 1010    LDWA 0x009A,s
1100 0110 0000 0000 1001 1010    LDWA 0x009A,sx
1100 1011 0000 0000 1001 1010    LDWX 0x009A,s
1100 1110 0000 0000 1001 1010
```

```
1100 0011 0000 0000 1001 1010    LDWA 0x009A,s
1100 0110 0000 0000 1001 1010    LDWA 0x009A,sx
1100 1011 0000 0000 1001 1010    LDWX 0x009A,s
1100 1110 0000 0000 1001 1010    LDWX 0x009A,sx
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.ASCII "Hello world!"
```

inserts the following string of 12 bytes:

```
48 65 6C 6C 6F 20 77 6F 72 6C 64 21
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.BYTE 43
.BYTE -3
.BYTE 0xF7
.BYTE 'H'
.BYTE 365
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.BYTE 43          inserts   2B
.BYTE -3
.BYTE 0xF7
.BYTE 'H'
.BYTE 365
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.BYTE 43          inserts   2B
.BYTE -3          inserts   FD
.BYTE 0xF7
.BYTE 'H'
.BYTE 365
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.BYTE 43        inserts  2B
.BYTE -3        inserts  FD
.BYTE 0xF7      inserts  F7
.BYTE 'H'
.BYTE 365
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.BYTE 43          inserts   2B
.BYTE -3          inserts   FD
.BYTE 0xF7        inserts   F7
.BYTE 'H'         inserts   48
.BYTE 365
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.BYTE 43        inserts  2B
.BYTE -3        inserts  FD
.BYTE 0xF7      inserts  F7
.BYTE 'H'       inserts  48
.BYTE 365       ERROR: Decimal constant is out of byte range
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.WORD 43
.WORD -3
.WORD 0xF7
.WORD 'H'
.WORD 365
.WORD 0xC42E
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.WORD 43          inserts   002B
.WORD -3
.WORD 0xF7
.WORD 'H'
.WORD 365
.WORD 0xC42E
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.WORD 43          inserts   002B
.WORD -3          inserts   FFFD
.WORD 0xF7
.WORD 'H'
.WORD 365
.WORD 0xC42E
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.WORD 43          inserts   002B
.WORD -3          inserts   FFFD
.WORD 0xF7        inserts   00F7
.WORD 'H'
.WORD 365
.WORD 0xC42E
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.WORD 43          inserts   002B
.WORD -3          inserts   FFFD
.WORD 0xF7        inserts   00F7
.WORD 'H'         inserts   0048
.WORD 365
.WORD 0xC42E
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.WORD 43          inserts   002B
.WORD -3          inserts   FFFD
.WORD 0xF7        inserts   00F7
.WORD 'H'         inserts   0048
.WORD 365         inserts   016D
.WORD 0xC42E
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.WORD 43          inserts  002B
.WORD -3          inserts  FFFD
.WORD 0xF7        inserts  00F7
.WORD 'H'         inserts  0048
.WORD 365         inserts  016D
.WORD 0xC42E      inserts  C42E
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.BLOCK 3     inserts
.BLOCK 4     inserts
.BLOCK 0xC   inserts
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.BLOCK 3    inserts 00 00 00
.BLOCK 4    inserts
.BLOCK 0xC  inserts
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.BLOCK 3    inserts 00 00 00
.BLOCK 4    inserts 00 00 00 00
.BLOCK 0xC  inserts
```

Pseudo-Operations

`.ASCII`	A string of ASCII bytes
`.BYTE`	A byte value
`.WORD`	A word value
`.BLOCK`	A block of bytes, all 0's

```
.BLOCK 3     inserts 00 00 00
.BLOCK 4     inserts 00 00 00 00
.BLOCK 0xC   inserts 00 00 00 00 00 00 00 00 00 00 00 00
```

Assembly Language Source Code

```
LDBA 0x000F,d     ;Load byte accumulator 'H'
STBA 0xFFFE,d     ;Store byte accumulator output port
ldba    0x0010  ,  D  ;Load byte accumulator 'i'
STBA 0xfffe,d     ;Store byte accumulator output port
STBA 0xFFFF,d     ;Store byte power off port
.ASCII "Hi"       ;ASCII "Hi" characters
```

Assembly Language Source Code

```
LDBA 0x000F,d     ;Load byte accumulator 'H'
STBA 0xFFFE,d     ;Store byte accumulator output port
ldba    0x0010  ,  D  ;Load byte accumulator 'i'
STBA 0xfffe,d     ;Store byte accumulator output port
STBA 0xFFFF,d     ;Store byte power off port
.ASCII "Hi"       ;ASCII "Hi" characters
```

Assembly Language Listing

```
-------------------------------------------------------------------------
      Object
Addr  code    Symbol   Mnemon  Operand     Comment
-------------------------------------------------------------------------
0000  D1000F           LDBA    0x000F,d    ;Load byte accumulator 'H'
0003  F1FFFE           STBA    0xFFFE,d    ;Store byte accumulator output port
0006  D10010           LDBA    0x0010,d    ;Load byte accumulator 'i'
0009  F1FFFE           STBA    0xFFFE,d    ;Store byte accumulator output port
000C  F1FFFF           STBA    0xFFFF,d    ;Store byte power off port
000F  4869             .ASCII  "Hi"        ;ASCII "Hi" characters
-------------------------------------------------------------------------
```

Assembly Language Source Code

```
LDBA 0x000F,d     ;Load byte accumulator 'H'
STBA 0xFFFE,d     ;Store byte accumulator output port
ldba    0x0010  ,  D  ;Load byte accumulator 'i'
STBA 0xfffe,d     ;Store byte accumulator output port
STBA 0xFFFF,d     ;Store byte power off port
.ASCII "Hi"       ;ASCII "Hi" characters
```

Assembly Language Listing

```
-------------------------------------------------------------------------
      Object
Addr  code    Symbol   Mnemon  Operand     Comment
-------------------------------------------------------------------------
0000  D1000F           LDBA    0x000F,d    ;Load byte accumulator 'H'
0003  F1FFFE           STBA    0xFFFE,d    ;Store byte accumulator output port
0006  D10010           LDBA    0x0010,d    ;Load byte accumulator 'i'
0009  F1FFFE           STBA    0xFFFE,d    ;Store byte accumulator output port
000C  F1FFFF           STBA    0xFFFF,d    ;Store byte power off port
000F  4869             .ASCII  "Hi"        ;ASCII "Hi" characters
-------------------------------------------------------------------------
```

Object Code

```
D1 00 0F F1 FF FE D1 00 10 F1 FF FE F1 FF FF 48
69
```

Demo Pepp IDE

Assembly Language Listing

```
0000  D1FFFD          LDBA    0xFFFD,d    ;Load first char from input port
                      STBA    0x0015,d    ;Store first char to 0015
                      LDBA    0xFFFD,d    ;Load from input port
                      STBA    0xFFFE,d    ;Store to output port
                      LDBA    0x0015,d    ;Load first char from 0015
                      STBA    0xFFFE,d    ;Store first char to output port
                      STBA    0xFFFF,d    ;Store to power off port
                      .BLOCK  1           ;One byte storage for first char
```

Assembly Language Listing

```
0000  D1FFFD          LDBA    0xFFFD,d    ;Load first char from input port
0003  F10015          STBA    0x0015,d    ;Store first char to 0015
                      LDBA    0xFFFD,d    ;Load from input port
                      STBA    0xFFFE,d    ;Store to output port
                      LDBA    0x0015,d    ;Load first char from 0015
                      STBA    0xFFFE,d    ;Store first char to output port
                      STBA    0xFFFF,d    ;Store to power off port
                      .BLOCK  1           ;One byte storage for first char
```

Assembly Language Listing

```
0000  D1FFFD          LDBA    0xFFFD,d    ;Load first char from input port
0003  F10015          STBA    0x0015,d    ;Store first char to 0015
0006  D1FFFD          LDBA    0xFFFD,d    ;Load from input port
                      STBA    0xFFFE,d    ;Store to output port
                      LDBA    0x0015,d    ;Load first char from 0015
                      STBA    0xFFFE,d    ;Store first char to output port
                      STBA    0xFFFF,d    ;Store to power off port
                      .BLOCK  1           ;One byte storage for first char
```

Assembly Language Listing

```
0000  D1FFFD          LDBA    0xFFFD,d    ;Load first char from input port
0003  F10015          STBA    0x0015,d    ;Store first char to 0015
0006  D1FFFD          LDBA    0xFFFD,d    ;Load from input port
0009  F1FFFE          STBA    0xFFFE,d    ;Store to output port
000C  D10015          LDBA    0x0015,d    ;Load first char from 0015
000F  F1FFFE          STBA    0xFFFE,d    ;Store first char to output port
0012  F1FFFF          STBA    0xFFFF,d    ;Store to power off port
                      .BLOCK  1           ;One byte storage for first char
```

Assembly Language Listing

```
0000  D1FFFD          LDBA    0xFFFD,d    ;Load first char from input port
0003  F10015          STBA    0x0015,d    ;Store first char to 0015
0006  D1FFFD          LDBA    0xFFFD,d    ;Load from input port
0009  F1FFFE          STBA    0xFFFE,d    ;Store to output port
000C  D10015          LDBA    0x0015,d    ;Load first char from 0015
000F  F1FFFE          STBA    0xFFFE,d    ;Store first char to output port
0012  F1FFFF          STBA    0xFFFF,d    ;Store to power off port
0015  00              .BLOCK  1           ;One byte storage for first char
```

Assembly Language Listing

```
0000   C1000F          LDWA    0x000F,d    ;Load 0005 from Mem[000F]
0003   510011          ADDA    0x0011,d    ;Add 0003 from Mem[0011]
0006   810013          ORA     0x0013,d    ;OR 0030 from Mem[0013]
                       STBA    0xFFFE,d    ;Store to output port
                       STBA    0xFFFF,d    ;Store to power off port
                       .WORD   5           ;Decimal 5
                       .WORD   3           ;Decimal 3
                       .WORD   0x0030      ;Mask for ASCII char
```

Assembly Language Listing

```
0000   C1000F          LDWA    0x000F,d    ;Load 0005 from Mem[000F]
0003   510011          ADDA    0x0011,d    ;Add 0003 from Mem[0011]
0006   810013          ORA     0x0013,d    ;OR 0030 from Mem[0013]
0009   F1FFFE          STBA    0xFFFE,d    ;Store to output port
                       STBA    0xFFFF,d    ;Store to power off port
                       .WORD   5           ;Decimal 5
                       .WORD   3           ;Decimal 3
                       .WORD   0x0030      ;Mask for ASCII char
```

Assembly Language Listing

```
0000   C1000F          LDWA    0x000F,d    ;Load 0005 from Mem[000F]
0003   510011          ADDA    0x0011,d    ;Add 0003 from Mem[0011]
0006   810013          ORA     0x0013,d    ;OR 0030 from Mem[0013]
0009   F1FFFE          STBA    0xFFFE,d    ;Store to output port
000C   F1FFFF          STBA    0xFFFF,d    ;Store to power off port
                .WORD   5           ;Decimal 5
                .WORD   3           ;Decimal 3
                .WORD   0x0030      ;Mask for ASCII char
```

Assembly Language Listing

```
0000  C1000F          LDWA    0x000F,d    ;Load 0005 from Mem[000F]
0003  510011          ADDA    0x0011,d    ;Add 0003 from Mem[0011]
0006  810013          ORA     0x0013,d    ;OR 0030 from Mem[0013]
0009  F1FFFE          STBA    0xFFFE,d    ;Store to output port
000C  F1FFFF          STBA    0xFFFF,d    ;Store to power off port
000F  0005            .WORD   5           ;Decimal 5
0011  0003            .WORD   3           ;Decimal 3
                      .WORD   0x0030      ;Mask for ASCII char
```

Assembly Language Listing

```
0000  C1000F          LDWA    0x000F,d    ;Load 0005 from Mem[000F]
0003  510011          ADDA    0x0011,d    ;Add 0003 from Mem[0011]
0006  810013          ORA     0x0013,d    ;OR 0030 from Mem[0013]
0009  F1FFFE          STBA    0xFFFE,d    ;Store to output port
000C  F1FFFF          STBA    0xFFFF,d    ;Store to power off port
000F  0005            .WORD   5           ;Decimal 5
0011  0003            .WORD   3           ;Decimal 3
0013  0030            .WORD   0x0030      ;Mask for ASCII char
```

Without symbols

```
0000  D1FFFD          LDBA    0xFFFD,d    ;Input first digit
0003  E10018          STWA    0x0018,d    ;Store to num
0006  D1FFFD          LDBA    0xFFFD,d    ;Input second digit
0009  510018          ADDA    0x0018,d    ;Add num to second digit
000C  71001A          ANDA    0x001A,d    ;Zero out the left nybble
000F  81001C          ORA     0x001C,d    ;Convert sum to ASCII character
0012  F1FFFE          STBA    0xFFFE,d    ;Output sum
0015  F1FFFF          STBA    0xFFFF,d    ;Shut down
0018  0000            .BLOCK  2
001A  000F            .WORD   0x000F
001C  0030            .WORD   0x0030
```

Without symbols

```
0000  D1FFFD        LDBA    0xFFFD,d    ;Input first digit
0003  E10018        STWA    0x0018,d    ;Store to num
0006  D1FFFD        LDBA    0xFFFD,d    ;Input second digit
0009  510018        ADDA    0x0018,d    ;Add num to second digit
000C  71001A        ANDA    0x001A,d    ;Zero out the left nybble
000F  81001C        ORA     0x001C,d    ;Convert sum to ASCII character
0012  F1FFFE        STBA    0xFFFE,d    ;Output sum
0015  F1FFFF        STBA    0xFFFF,d    ;Shut down
0018  0000          .BLOCK  2
001A  000F          .WORD   0x000F
001C  0030          .WORD   0x0030
```

Without symbols

```
0000  D1FFFD          LDBA    0xFFFD,d    ;Input first digit
0003  E10018          STWA    0x0018,d    ;Store to num
0006  D1FFFD          LDBA    0xFFFD,d    ;Input second digit
0009  510018          ADDA    0x0018,d    ;Add num to second digit
000C  71001A          ANDA    0x001A,d    ;Zero out the left nybble
000F  81001C          ORA     0x001C,d    ;Convert sum to ASCII character
0012  F1FFFE          STBA    0xFFFE,d    ;Output sum
0015  F1FFFF          STBA    0xFFFF,d    ;Shut down
0018  0000            .BLOCK  2
001A  000F            .WORD   0x000F
001C  0030            .WORD   0x0030
```

With symbols

```
0000  D1FFFD          LDBA    charIn,d    ;Input first digit
0003  E10018          STWA    num,d       ;Store to num
0006  D1FFFD          LDBA    charIn,d    ;Input second digit
0009  510018          ADDA    num,d       ;Add num to second digit
000C  71001A          ANDA    andMask,d   ;Zero out the left nybble
000F  81001C          ORA     orMask,d    ;Convert sum to ASCII character
0012  F1FFFE          STBA    charOut,d   ;Output sum
0015  F1FFFF          STBA    pwrOff,d    ;Shut down
0018  0000    num:    .BLOCK  2
001A  000F    andMask: .WORD  0x000F
001C  0030    orMask: .WORD   0x0030
```

Symbols

- Defined by an identifier followed by a colon at the start of a statement
- The value of a symbol is the address of the object code generated by the statement

Assembly Language Listing

```
0000  D1FFFD           LDBA    charIn,d    ;Input first digit
0003  E10018           STWA    num,d       ;Store to num
0006  D1FFFD           LDBA    charIn,d    ;Input second digit
0009  510018           ADDA    num,d       ;Add num to second digit
000C  71001A           ANDA    andMask,d   ;Zero out the left nybble
000F  81001C           ORA     orMask,d    ;Convert sum to ASCII character
0012  F1FFFE           STBA    charOut,d   ;Output sum
0015  F1FFFF           STBA    pwrOff,d    ;Shut down
0018  0000    num:     .BLOCK  2
001A  000F    andMask: .WORD   0x000F
001C  0030    orMask:  .WORD   0x0030
```

(a) The program.

Symbol	Value	Symbol	Value
num	0018	charIn	
andMask		charOut	
orMask		pwrOff	

(b) The symbol table.

Assembly Language Listing

```
0000  D1FFFD          LDBA    charIn,d    ;Input first digit
0003  E10018          STWA    num,d       ;Store to num
0006  D1FFFD          LDBA    charIn,d    ;Input second digit
0009  510018          ADDA    num,d       ;Add num to second digit
000C  71001A          ANDA    andMask,d   ;Zero out the left nybble
000F  81001C          ORA     orMask,d    ;Convert sum to ASCII character
0012  F1FFFE          STBA    charOut,d   ;Output sum
0015  F1FFFF          STBA    pwrOff,d    ;Shut down
0018  0000    num:    .BLOCK  2
001A  000F    andMask: .WORD   0x000F
001C  0030    orMask:  .WORD   0x0030
```

(a) The program.

Symbol	Value	Symbol	Value
num	0018	charIn	
andMask	001A	charOut	
orMask		pwrOff	

(b) The symbol table.

Assembly Language Listing

```
0000  D1FFFD          LDBA    charIn,d    ;Input first digit
0003  E10018          STWA    num,d       ;Store to num
0006  D1FFFD          LDBA    charIn,d    ;Input second digit
0009  510018          ADDA    num,d       ;Add num to second digit
000C  71001A          ANDA    andMask,d   ;Zero out the left nybble
000F  81001C          ORA     orMask,d    ;Convert sum to ASCII character
0012  F1FFFE          STBA    charOut,d   ;Output sum
0015  F1FFFF          STBA    pwrOff,d    ;Shut down
0018  0000    num:    .BLOCK  2
001A  000F    andMask: .WORD   0x000F
001C  0030    orMask:  .WORD   0x0030
```

(a) The program.

Symbol	Value	Symbol	Value
num	0018	charIn	
andMask	001A	charOut	
orMask	001C	pwrOff	

(b) The symbol table.

Assembly Language Listing

```
0000  D1FFFD          LDBA    charIn,d    ;Input first digit
0003  E10018          STWA    num,d       ;Store to num
0006  D1FFFD          LDBA    charIn,d    ;Input second digit
0009  510018          ADDA    num,d       ;Add num to second digit
000C  71001A          ANDA    andMask,d   ;Zero out the left nybble
000F  81001C          ORA     orMask,d    ;Convert sum to ASCII character
0012  F1FFFE          STBA    charOut,d   ;Output sum
0015  F1FFFF          STBA    pwrOff,d    ;Shut down
0018  0000    num:    .BLOCK  2
001A  000F    andMask: .WORD   0x000F
001C  0030    orMask:  .WORD   0x0030
```

(a) The program.

Symbol	Value	Symbol	Value
`num`	0018	`charIn`	FFFD
`andMask`	001A	`charOut`	
`orMask`	001C	`pwrOff`	

(b) The symbol table.

Assembly Language Listing

```
0000  D1FFFD          LDBA    charIn,d    ;Input first digit
0003  E10018          STWA    num,d       ;Store to num
0006  D1FFFD          LDBA    charIn,d    ;Input second digit
0009  510018          ADDA    num,d       ;Add num to second digit
000C  71001A          ANDA    andMask,d   ;Zero out the left nybble
000F  81001C          ORA     orMask,d    ;Convert sum to ASCII character
0012  F1FFFE          STBA    charOut,d   ;Output sum
0015  F1FFFF          STBA    pwrOff,d    ;Shut down
0018  0000    num:    .BLOCK  2
001A  000F    andMask: .WORD   0x000F
001C  0030    orMask:  .WORD   0x0030
```

(a) The program.

Symbol	Value	Symbol	Value
`num`	0018	`charIn`	FFFD
`andMask`	001A	`charOut`	FFFE
`orMask`	001C	`pwrOff`	

(b) The symbol table.

Assembly Language Listing

```
0000  D1FFFD          LDBA    charIn,d    ;Input first digit
0003  E10018          STWA    num,d       ;Store to num
0006  D1FFFD          LDBA    charIn,d    ;Input second digit
0009  510018          ADDA    num,d       ;Add num to second digit
000C  71001A          ANDA    andMask,d   ;Zero out the left nybble
000F  81001C          ORA     orMask,d    ;Convert sum to ASCII character
0012  F1FFFE          STBA    charOut,d   ;Output sum
0015  F1FFFF          STBA    pwrOff,d    ;Shut down
0018  0000    num:    .BLOCK  2
001A  000F    andMask: .WORD   0x000F
001C  0030    orMask:  .WORD   0x0030
```

(a) The program.

Symbol	Value	Symbol	Value
`num`	0018	`charIn`	FFFD
`andMask`	001A	`charOut`	FFFE
`orMask`	001C	`pwrOff`	FFFF

(b) The symbol table.

Demo Pepp IDE

Assembly language macros

```
0000  D1FFFD          LDBA    charIn,d    ;Load first char from input port
0003  F10015          STBA    char_1,d    ;Store first char to char_1
0006  D1FFFD          LDBA    charIn,d    ;Load from input port
0009  F1FFFE          STBA    charOut,d   ;Store to output port
000C  D10015          LDBA    char_1,d    ;Load first char from char_1
000F  F1FFFE          STBA    charOut,d   ;Store first char to output port
0012  F1FFFF          STBA    pwrOff,d    ;Store to power off port
0015  00      char_1: .BLOCK  1           ;One byte storage for first char
```

Assembly language macros

```
0000  D1FFFD          LDBA    charIn,d    ;Load first char from input port
0003  F10015          STBA    char_1,d    ;Store first char to char_1
0006  D1FFFD          LDBA    charIn,d    ;Load from input port
0009  F1FFFE          STBA    charOut,d   ;Store to output port
000C  D10015          LDBA    char_1,d    ;Load first char from char_1
000F  F1FFFE          STBA    charOut,d   ;Store first char to output port
0012  F1FFFF          STBA    pwrOff,d    ;Store to power off port
0015  00      char_1: .BLOCK  1           ;One byte storage for first char
```

Common: Input a char from `charIn` to memory.

Assembly language macros

```
0000  D1FFFD          LDBA    charIn,d    ;Load first char from input port
0003  F10015          STBA    char_1,d    ;Store first char to char_1
0006  D1FFFD          LDBA    charIn,d    ;Load from input port
0009  F1FFFE          STBA    charOut,d   ;Store to output port
000C  D10015          LDBA    char_1,d    ;Load first char from char_1
000F  F1FFFE          STBA    charOut,d   ;Store first char to output port
0012  F1FFFF          STBA    pwrOff,d    ;Store to power off port
0015  00      char_1: .BLOCK  1           ;One byte storage for first char
```

Common: Input a char from `charIn` to memory.

Common: Output a char from memory to `charOut`.

Assembly Language Source Code

```
          @CHARI   char_1,d     ;Input first char to char_1
          LDBA     charIn,d     ;Load from input port
          STBA     charOut,d    ;Store to output port
          @CHARO   char_1,d     ;Output char_1
          STBA     pwrOff,d     ;Store to power off port
char_1:   .BLOCK   1            ;One byte storage for first char
```

Assembly Language Source Code

```
          @CHARI   char_1,d     ;Input first char to char_1
          LDBA     charIn,d     ;Load from input port
          STBA     charOut,d    ;Store to output port
          @CHARO   char_1,d     ;Output char_1
          STBA     pwrOff,d     ;Store to power off port
char_1:   .BLOCK   1            ;One byte storage for first char
```

Assembly Language Listing

```
                ;        @CHARI   char_1,d     ;Input first char to char_1
0000  D1FFFD             LDBA     charIn,d
0003  F10015             STBA     char_1,d
                ;        End @CHARI
0006  D1FFFD             LDBA     charIn,d     ;Load from input port
0009  F1FFFE             STBA     charOut,d    ;Store to output port
                ;        @CHARO   char_1,d     ;Output char_1
000C  D10015             LDBA     char_1,d
000F  F1FFFE             STBA     charOut,d
                ;        End @CHARO
0012  F1FFFF             STBA     pwrOff,d     ;Store to power off port
0015  00      char_1:    .BLOCK   1            ;One byte storage for first char
```

The `@CHARI` and `@CHARO` macro expansions

Operation	Macro	Macro Expansion
Character input	`@CHARI` OprndSpec,AddrMode	`LDBA charIn,d` `STBA` OprndSpec,AddrMode
Character output	`@CHARO` OprndSpec,AddrMode	`LDBA` OprndSpec,AddrMode `STBA charOut,d`

Direct addressing

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

Immediate addressing

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

Assembly Language Source Code

```
@CHARO   'H',i          ;Output the H character
@CHARO   'i',i          ;Output the i character
STBA     pwrOff,d       ;Store to power off port
```

Assembly Language Source Code

```
@CHARO   'H',i          ;Output the H character
@CHARO   'i',i          ;Output the i character
STBA     pwrOff,d       ;Store to power off port
```

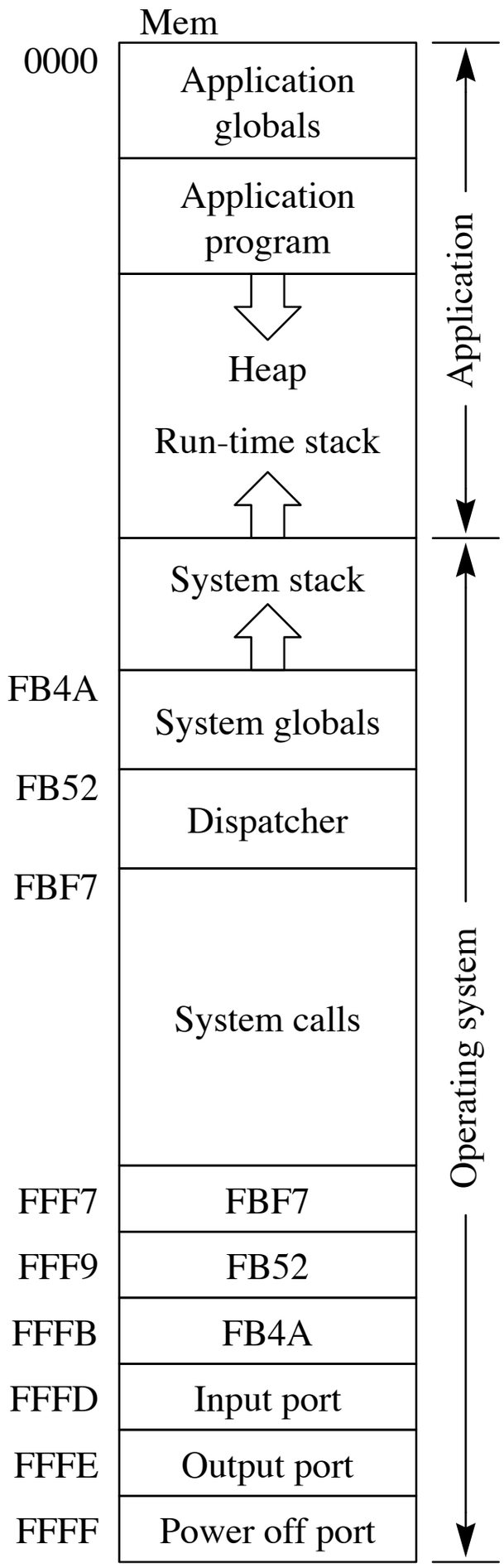
Assembly Language Listing

```
                ;       @CHARO   'H',i          ;Output the H character
0000  D00048            LDBA     'H',i
0003  F1FFFE            STBA     charOut,d
                ;       End @CHARO
                ;       @CHARO   'i',i          ;Output the i character
0006  D00069            LDBA     'i',i
0009  F1FFFE            STBA     charOut,d
                ;       End @CHARO
000C  F1FFFF            STBA     pwrOff,d       ;Store to power off port
```

An Assembler at Level 5

- Can use the services of an operating system
- The operating system calls the app from the dispatcher
- The app calls the operating system with the `SCALL` trap instruction

Figure 5.11



The Pep/10 memory map in full OS mode

Level Asmb3

Assembly Language Source Code

```
@CHARO   'H',i          ;Output the H character
@CHARO   'i',i          ;Output the i character
STBA     pwrOff,d       ;Store to power off port
```

Level Asmb5

Assembly Language Source Code

```
@CHARO   'H',i          ;Output the H character
@CHARO   'i',i          ;Output the i character
RET
```

The system call trap instruction
`SCALL`

The system call trap instruction `SCALL`

- `DECI` Decimal input
- `DECO` Decimal output
- `HEXO` Hexadecimal output
- `STRO` String output
- `SNOP` System call no operation

The system call trap instruction `SCALL`

- `DECI` Decimal input
- `DECO` Decimal output
- `HEXO` Hexadecimal output
- `STRO` String output
- `SNOP` System call no operation

Symbol	Value
`DECI`	0000
`DECO`	0001
`HEXO`	0002
`STRO`	0003
`SNOP`	0004

The system call trap protocol

- Load the symbol for your system call into the accumulator using immediate addressing.
- Execute `SCALL` with an appropriate operand specifier and addressing mode of your choice.

Assembly Language Source Code

```
            LDWA    STRO,i
            SCALL   msg,d
            RET
msg:        .ASCII  "Hello, world!\n\0"
```

Output

```
Hello, world!
```

Assembly Language Source Code

```
            LDWA    STRO,i
            SCALL   msg,d
            RET
msg:        .ASCII  "Hello, world!\n\0"
```

Output

```
Hello, world!
```

Assembly Language Source Code

```
            LDWA    STRO,i
            SCALL   msg,d
            RET
msg:        .ASCII  "Hello, world!\n\0"
```

Assembly Language Listing

```
0000  C00003          LDWA    STRO,i
0003  390007          SCALL   msg,d
0006  01              RET
0007  48656C msg:     .ASCII  "Hello, world!\n\0"
      6C6F2C
      20776F
      726C64
      210A00
```

Output

```
Hello, world!
```

The process control block (PCB)

The process control block (PCB)

Created by the `SCALL` instruction.

The process control block (PCB)

Created by the `SCALL` instruction.

Contains a copy of all the registers in the CPU.

The process control block (PCB)

Created by the `SCALL` instruction.

Contains a copy of all the registers in the CPU.

```
0000   C00003          LDWA     STRO,i
0003   390007          SCALL    msg,d
0006   01              RET
0007   48656C msg:     .ASCII   "Hello, world!\n\0"
```

The process control block (PCB)

Created by the `SCALL` instruction.

Contains a copy of all the registers in the CPU.

```
0000   C00003          LDWA     STRO,i
0003   390007          SCALL    msg,d
0006   01              RET
0007   48656C msg:     .ASCII   "Hello, world!\n\0"
```

The process control block (PCB)

Created by the `SCALL` instruction.

Contains a copy of all the registers in the CPU.

```
0000   C00003          LDWA     STRO,i
0003   390007          SCALL    msg,d
0006   01              RET
0007   48656C msg:     .ASCII   "Hello, world!\n\0"
```

Fetch:

The process control block (PCB)

Created by the `SCALL` instruction.

Contains a copy of all the registers in the CPU.

```
0000   C00003          LDWA     STRO,i
0003   390007          SCALL    msg,d
0006   01              RET
0007   48656C msg:     .ASCII   "Hello, world!\n\0"
```

Fetch: IR ← instruction

The process control block (PCB)

Created by the `SCALL` instruction.

Contains a copy of all the registers in the CPU.

```
0000   C00003          LDWA     STRO,i
0003   390007          SCALL    msg,d
0006   01              RET
0007   48656C msg:     .ASCII   "Hello, world!\n\0"
```

Fetch: IR ← instruction
Decode

The process control block (PCB)

Created by the `SCALL` instruction.

Contains a copy of all the registers in the CPU.

```
0000   C00003          LDWA     STRO,i
0003   390007          SCALL    msg,d
0006   01              RET
0007   48656C msg:     .ASCII   "Hello, world!\n\0"
```

Fetch: IR ← instruction
Decode
Increment:

The process control block (PCB)

Created by the `SCALL` instruction.

Contains a copy of all the registers in the CPU.

```
0000   C00003          LDWA     STRO,i
0003   390007          SCALL    msg,d
0006   01              RET
0007   48656C msg:     .ASCII   "Hello, world!\n\0"
```

Fetch: IR ← instruction

Decode

Increment: PC ← PC + 3

The process control block (PCB)

Created by the `SCALL` instruction.

Contains a copy of all the registers in the CPU.

```
0000   C00003          LDWA     STRO,i
0003   390007          SCALL    msg,d
0006   01              RET
0007   48656C msg:     .ASCII   "Hello, world!\n\0"
```

Fetch: IR ← instruction

Decode

Increment: PC ← PC + 3

Execute:

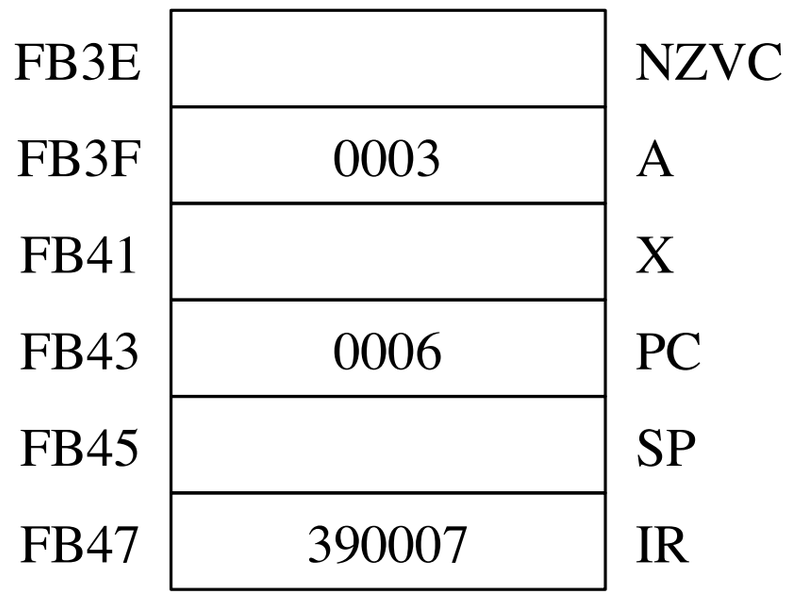
The process control block (PCB)

Created by the `SCALL` instruction.

Contains a copy of all the registers in the CPU.

```
0000   C00003          LDWA     STRO,i
0003   390007          SCALL    msg,d
0006   01              RET
0007   48656C msg:     .ASCII   "Hello, world!\n\0"
```

Fetch: IR ← instruction
Decode
Increment: PC ← PC + 3
Execute:



The `DECI` and `DECO` system calls

The DECI and DECO system calls

```
0047   0000    width:    .BLOCK  2
0049   0000    height:   .BLOCK  2
004B   0000    perim:    .BLOCK  2
004D   776964 msg1:     .ASCII  "width: \0"
       74683A
       2000
0055   686569 msg2:     .ASCII  "height: \0"
       676874
       3A2000
005E   706572 msg3:     .ASCII  "perimeter: \0"
       696D65
       746572
       3A2000
```

Input

```
24  17
```

Output

```
width: 24
height: 17
perimeter: 82
```

The DECI and DECO system calls

```
0047   0000    width:    .BLOCK   2
0049   0000    height:   .BLOCK   2
004B   0000    perim:    .BLOCK   2
004D   776964 msg1:      .ASCII   "width: \0"
       74683A
       2000
0055   686569 msg2:      .ASCII   "height: \0"
       676874
       3A2000
005E   706572 msg3:      .ASCII   "perimeter: \0"
       696D65
       746572
       3A2000
```

Input

```
24  17
```

Output

```
width: 24
height: 17
perimeter: 82
```

Symbol	Value
width	
height	
perim	
msg1	
msg2	
msg3	

The DECI and DECO system calls

```
0047   0000    width:    .BLOCK  2
0049   0000    height:   .BLOCK  2
004B   0000    perim:    .BLOCK  2
004D   776964 msg1:     .ASCII  "width: \0"
       74683A
       2000
0055   686569 msg2:     .ASCII  "height: \0"
       676874
       3A2000
005E   706572 msg3:     .ASCII  "perimeter: \0"
       696D65
       746572
       3A2000
```

Input

```
24  17
```

Output

```
width: 24
height: 17
perimeter: 82
```

Symbol	Value
width	0047
height	
perim	
msg1	
msg2	
msg3	

The DECI and DECO system calls

```
0047   0000    width:    .BLOCK  2
0049   0000    height:   .BLOCK  2
004B   0000    perim:    .BLOCK  2
004D   776964 msg1:     .ASCII  "width: \0"
       74683A
       2000
0055   686569 msg2:     .ASCII  "height: \0"
       676874
       3A2000
005E   706572 msg3:     .ASCII  "perimeter: \0"
       696D65
       746572
       3A2000
```

Input

```
24  17
```

Output

```
width: 24
height: 17
perimeter: 82
```

Symbol	Value
width	0047
height	0049
perim	
msg1	
msg2	
msg3	

The DECI and DECO system calls

```
0047   0000    width:    .BLOCK   2
0049   0000    height:   .BLOCK   2
004B   0000    perim:    .BLOCK   2
004D   776964 msg1:      .ASCII   "width: \0"
       74683A
       2000
0055   686569 msg2:      .ASCII   "height: \0"
       676874
       3A2000
005E   706572 msg3:      .ASCII   "perimeter: \0"
       696D65
       746572
       3A2000
```

Input

```
24  17
```

Output

```
width: 24
height: 17
perimeter: 82
```

Symbol	Value
width	0047
height	0049
perim	004B
msg1	
msg2	
msg3	

The DECI and DECO system calls

```
0047   0000    width:    .BLOCK   2
0049   0000    height:   .BLOCK   2
004B   0000    perim:    .BLOCK   2
004D   776964 msg1:      .ASCII   "width: \0"
       74683A
       2000
0055   686569 msg2:      .ASCII   "height: \0"
       676874
       3A2000
005E   706572 msg3:      .ASCII   "perimeter: \0"
       696D65
       746572
       3A2000
```

Input

```
24  17
```

Output

```
width: 24
height: 17
perimeter: 82
```

Symbol	Value
width	0047
height	0049
perim	004B
msg1	004D
msg2	
msg3	

The DECI and DECO system calls

```
0047   0000    width:    .BLOCK   2
0049   0000    height:   .BLOCK   2
004B   0000    perim:    .BLOCK   2
004D   776964 msg1:      .ASCII   "width: \0"
       74683A
       2000
0055   686569 msg2:      .ASCII   "height: \0"
       676874
       3A2000
005E   706572 msg3:      .ASCII   "perimeter: \0"
       696D65
       746572
       3A2000
```

Input

```
24  17
```

Output

```
width: 24
height: 17
perimeter: 82
```

Symbol	Value
width	0047
height	0049
perim	004B
msg1	004D
msg2	0055
msg3	

The DECI and DECO system calls

```
0047   0000    width:    .BLOCK  2
0049   0000    height:   .BLOCK  2
004B   0000    perim:    .BLOCK  2
004D   776964 msg1:     .ASCII  "width: \0"
       74683A
       2000
0055   686569 msg2:     .ASCII  "height: \0"
       676874
       3A2000
005E   706572 msg3:     .ASCII  "perimeter: \0"
       696D65
       746572
       3A2000
```

Input

```
24  17
```

Output

```
width: 24
height: 17
perimeter: 82
```

Symbol	Value
width	0047
height	0049
perim	004B
msg1	004D
msg2	0055
msg3	005E

```
;Input width

;Input height

;Compute perimeter of rectangle



;Output "width: "

;Output width

;Output newline character

;Output "height: "

;Output height

;Output newline character

;Output "perimeter: "

;Output perimeter
```

```
0000  C00000          LDWA    DECI,i      ;Input width

                                          ;Input height

                                          ;Compute perimeter of rectangle



                                          ;Output "width: "

                                          ;Output width

                                          ;Output newline character

                                          ;Output "height: "

                                          ;Output height

                                          ;Output newline character

                                          ;Output "perimeter: "

                                          ;Output perimeter
```

```
0000  C00000          LDWA    DECI,i      ;Input width
0003  390047          SCALL   width,d
                                          ;Input height

                                          ;Compute perimeter of rectangle



                                          ;Output "width: "

                                          ;Output width

                                          ;Output newline character

                                          ;Output "height: "

                                          ;Output height

                                          ;Output newline character

                                          ;Output "perimeter: "

                                          ;Output perimeter
```

```
0000  C00000          LDWA    DECI,i      ;Input width
0003  390047          SCALL   width,d
0006  C00000          LDWA    DECI,i      ;Input height

                                          ;Compute perimeter of rectangle



                                          ;Output "width: "

                                          ;Output width

                                          ;Output newline character

                                          ;Output "height: "

                                          ;Output height

                                          ;Output newline character

                                          ;Output "perimeter: "

                                          ;Output perimeter
```

```
0000  C00000          LDWA    DECI,i      ;Input width
0003  390047          SCALL   width,d
0006  C00000          LDWA    DECI,i      ;Input height
0009  390049          SCALL   height,d
                                          ;Compute perimeter of rectangle



                                          ;Output "width: "

                                          ;Output width

                                          ;Output newline character

                                          ;Output "height: "

                                          ;Output height

                                          ;Output newline character

                                          ;Output "perimeter: "

                                          ;Output perimeter
```

```
0000  C00000          LDWA    DECI,i      ;Input width
0003  390047          SCALL   width,d
0006  C00000          LDWA    DECI,i      ;Input height
0009  390049          SCALL   height,d
000C  C10047          LDWA    width,d     ;Compute perimeter of rectangle


                                          ;Output "width: "

                                          ;Output width

                                          ;Output newline character

                                          ;Output "height: "

                                          ;Output height

                                          ;Output newline character

                                          ;Output "perimeter: "

                                          ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d


                                        ;Output "width: "

                                        ;Output width

                                        ;Output newline character

                                        ;Output "height: "

                                        ;Output height

                                        ;Output newline character

                                        ;Output "perimeter: "

                                        ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA


                                        ;Output "width: "

                                        ;Output width

                                        ;Output newline character

                                        ;Output "height: "

                                        ;Output height

                                        ;Output newline character

                                        ;Output "perimeter: "

                                        ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d

                                        ;Output "width: "

                                        ;Output width

                                        ;Output newline character

                                        ;Output "height: "

                                        ;Output height

                                        ;Output newline character

                                        ;Output "perimeter: "

                                        ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "

                                        ;Output width

                                        ;Output newline character

                                        ;Output "height: "

                                        ;Output height

                                        ;Output newline character

                                        ;Output "perimeter: "

                                        ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "
0019  39004D        SCALL   msg1,d
                                        ;Output width
                                        ;Output newline character
                                        ;Output "height: "
                                        ;Output height
                                        ;Output newline character
                                        ;Output "perimeter: "
                                        ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "
0019  39004D        SCALL   msg1,d
001C  C00001        LDWA    DECO,i      ;Output width
```

```
                    ;Output newline character
                    ;Output "height: "
                    ;Output height
                    ;Output newline character
                    ;Output "perimeter: "
                    ;Output perimeter
```

```
0000  C00000          LDWA    DECI,i      ;Input width
0003  390047          SCALL   width,d
0006  C00000          LDWA    DECI,i      ;Input height
0009  390049          SCALL   height,d
000C  C10047          LDWA    width,d     ;Compute perimeter of rectangle
000F  510049          ADDA    height,d
0012  1A              ASLA
0013  E1004B          STWA    perim,d
0016  C00003          LDWA    STRO,i      ;Output "width: "
0019  39004D          SCALL   msg1,d
001C  C00001          LDWA    DECO,i      ;Output width
001F  390047          SCALL   width,d
                                          ;Output newline character
                                          ;Output "height: "
                                          ;Output height
                                          ;Output newline character
                                          ;Output "perimeter: "
                                          ;Output perimeter
```

```
0000  C00000          LDWA    DECI,i      ;Input width
0003  390047          SCALL   width,d
0006  C00000          LDWA    DECI,i      ;Input height
0009  390049          SCALL   height,d
000C  C10047          LDWA    width,d     ;Compute perimeter of rectangle
000F  510049          ADDA    height,d
0012  1A              ASLA
0013  E1004B          STWA    perim,d
0016  C00003          LDWA    STRO,i      ;Output "width: "
0019  39004D          SCALL   msg1,d
001C  C00001          LDWA    DECO,i      ;Output width
001F  390047          SCALL   width,d
0022  D0000A          LDBA    '\n',i      ;Output newline character

                                          ;Output "height: "

                                          ;Output height

                                          ;Output newline character

                                          ;Output "perimeter: "

                                          ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "
0019  39004D        SCALL   msg1,d
001C  C00001        LDWA    DECO,i      ;Output width
001F  390047        SCALL   width,d
0022  D0000A        LDBA    '\n',i      ;Output newline character
0025  F1FFFE        STBA    charOut,d
                                        ;Output "height: "

                                        ;Output height

                                        ;Output newline character

                                        ;Output "perimeter: "

                                        ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "
0019  39004D        SCALL   msg1,d
001C  C00001        LDWA    DECO,i      ;Output width
001F  390047        SCALL   width,d
0022  D0000A        LDBA    '\n',i      ;Output newline character
0025  F1FFFE        STBA    charOut,d
0028  C00003        LDWA    STRO,i      ;Output "height: "

                                        ;Output height

                                        ;Output newline character

                                        ;Output "perimeter: "

                                        ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "
0019  39004D        SCALL   msg1,d
001C  C00001        LDWA    DECO,i      ;Output width
001F  390047        SCALL   width,d
0022  D0000A        LDBA    '\n',i      ;Output newline character
0025  F1FFFE        STBA    charOut,d
0028  C00003        LDWA    STRO,i      ;Output "height: "
002B  390055        SCALL   msg2,d
                                        ;Output height

                                        ;Output newline character

                                        ;Output "perimeter: "

                                        ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "
0019  39004D        SCALL   msg1,d
001C  C00001        LDWA    DECO,i      ;Output width
001F  390047        SCALL   width,d
0022  D0000A        LDBA    '\n',i      ;Output newline character
0025  F1FFFE        STBA    charOut,d
0028  C00003        LDWA    STRO,i      ;Output "height: "
002B  390055        SCALL   msg2,d
002E  C00001        LDWA    DECO,i      ;Output height

                                        ;Output newline character

                                        ;Output "perimeter: "

                                        ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "
0019  39004D        SCALL   msg1,d
001C  C00001        LDWA    DECO,i      ;Output width
001F  390047        SCALL   width,d
0022  D0000A        LDBA    '\n',i      ;Output newline character
0025  F1FFFE        STBA    charOut,d
0028  C00003        LDWA    STRO,i      ;Output "height: "
002B  390055        SCALL   msg2,d
002E  C00001        LDWA    DECO,i      ;Output height
0031  390049        SCALL   height,d
                                        ;Output newline character

                                        ;Output "perimeter: "

                                        ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "
0019  39004D        SCALL   msg1,d
001C  C00001        LDWA    DECO,i      ;Output width
001F  390047        SCALL   width,d
0022  D0000A        LDBA    '\n',i      ;Output newline character
0025  F1FFFE        STBA    charOut,d
0028  C00003        LDWA    STRO,i      ;Output "height: "
002B  390055        SCALL   msg2,d
002E  C00001        LDWA    DECO,i      ;Output height
0031  390049        SCALL   height,d
0034  D0000A        LDBA    '\n',i      ;Output newline character

                                        ;Output "perimeter: "

                                        ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "
0019  39004D        SCALL   msg1,d
001C  C00001        LDWA    DECO,i      ;Output width
001F  390047        SCALL   width,d
0022  D0000A        LDBA    '\n',i      ;Output newline character
0025  F1FFFE        STBA    charOut,d
0028  C00003        LDWA    STRO,i      ;Output "height: "
002B  390055        SCALL   msg2,d
002E  C00001        LDWA    DECO,i      ;Output height
0031  390049        SCALL   height,d
0034  D0000A        LDBA    '\n',i      ;Output newline character
0037  F1FFFE        STBA    charOut,d
                                        ;Output "perimeter: "

                                        ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "
0019  39004D        SCALL   msg1,d
001C  C00001        LDWA    DECO,i      ;Output width
001F  390047        SCALL   width,d
0022  D0000A        LDBA    '\n',i      ;Output newline character
0025  F1FFFE        STBA    charOut,d
0028  C00003        LDWA    STRO,i      ;Output "height: "
002B  390055        SCALL   msg2,d
002E  C00001        LDWA    DECO,i      ;Output height
0031  390049        SCALL   height,d
0034  D0000A        LDBA    '\n',i      ;Output newline character
0037  F1FFFE        STBA    charOut,d
003A  C00003        LDWA    STRO,i      ;Output "perimeter: "

                                        ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "
0019  39004D        SCALL   msg1,d
001C  C00001        LDWA    DECO,i      ;Output width
001F  390047        SCALL   width,d
0022  D0000A        LDBA    '\n',i      ;Output newline character
0025  F1FFFE        STBA    charOut,d
0028  C00003        LDWA    STRO,i      ;Output "height: "
002B  390055        SCALL   msg2,d
002E  C00001        LDWA    DECO,i      ;Output height
0031  390049        SCALL   height,d
0034  D0000A        LDBA    '\n',i      ;Output newline character
0037  F1FFFE        STBA    charOut,d
003A  C00003        LDWA    STRO,i      ;Output "perimeter: "
003D  39005E        SCALL   msg3,d
                                        ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "
0019  39004D        SCALL   msg1,d
001C  C00001        LDWA    DECO,i      ;Output width
001F  390047        SCALL   width,d
0022  D0000A        LDBA    '\n',i      ;Output newline character
0025  F1FFFE        STBA    charOut,d
0028  C00003        LDWA    STRO,i      ;Output "height: "
002B  390055        SCALL   msg2,d
002E  C00001        LDWA    DECO,i      ;Output height
0031  390049        SCALL   height,d
0034  D0000A        LDBA    '\n',i      ;Output newline character
0037  F1FFFE        STBA    charOut,d
003A  C00003        LDWA    STRO,i      ;Output "perimeter: "
003D  39005E        SCALL   msg3,d
0040  C00001        LDWA    DECO,i      ;Output perimeter
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "
0019  39004D        SCALL   msg1,d
001C  C00001        LDWA    DECO,i      ;Output width
001F  390047        SCALL   width,d
0022  D0000A        LDBA    '\n',i      ;Output newline character
0025  F1FFFE        STBA    charOut,d
0028  C00003        LDWA    STRO,i      ;Output "height: "
002B  390055        SCALL   msg2,d
002E  C00001        LDWA    DECO,i      ;Output height
0031  390049        SCALL   height,d
0034  D0000A        LDBA    '\n',i      ;Output newline character
0037  F1FFFE        STBA    charOut,d
003A  C00003        LDWA    STRO,i      ;Output "perimeter: "
003D  39005E        SCALL   msg3,d
0040  C00001        LDWA    DECO,i      ;Output perimeter
0043  39004B        SCALL   perim,d
```

```
0000  C00000        LDWA    DECI,i      ;Input width
0003  390047        SCALL   width,d
0006  C00000        LDWA    DECI,i      ;Input height
0009  390049        SCALL   height,d
000C  C10047        LDWA    width,d     ;Compute perimeter of rectangle
000F  510049        ADDA    height,d
0012  1A            ASLA
0013  E1004B        STWA    perim,d
0016  C00003        LDWA    STRO,i      ;Output "width: "
0019  39004D        SCALL   msg1,d
001C  C00001        LDWA    DECO,i      ;Output width
001F  390047        SCALL   width,d
0022  D0000A        LDBA    '\n',i      ;Output newline character
0025  F1FFFE        STBA    charOut,d
0028  C00003        LDWA    STRO,i      ;Output "height: "
002B  390055        SCALL   msg2,d
002E  C00001        LDWA    DECO,i      ;Output height
0031  390049        SCALL   height,d
0034  D0000A        LDBA    '\n',i      ;Output newline character
0037  F1FFFE        STBA    charOut,d
003A  C00003        LDWA    STRO,i      ;Output "perimeter: "
003D  39005E        SCALL   msg3,d
0040  C00001        LDWA    DECO,i      ;Output perimeter
0043  39004B        SCALL   perim,d
0046  01            RET
```

The HEXO system call

The HEXO system call

Assembly Language Listing

```
0013   FFFD    num:        .WORD    -3
```

Output

The HEXO system call

Assembly Language Listing

```
0000   C00001              LDWA     DECO,i
```

```
0013   FFFD    num:       .WORD    -3
```

Output

The HEXO system call

Assembly Language Listing

```
0000   C00001          LDWA    DECO,i
0003   390013          SCALL   num,d

0013   FFFD    num:    .WORD   -3
```

Output

```
-3
```

The HEXO system call

Assembly Language Listing

```
0000   C00001          LDWA    DECO,i
0003   390013          SCALL   num,d
0006   D00020          LDBA    ' ',i
```

```
0013   FFFD    num:    .WORD   -3
```

Output

```
-3
```

The HEXO system call

Assembly Language Listing

```
0000   C00001          LDWA    DECO,i
0003   390013          SCALL   num,d
0006   D00020          LDBA    ' ',i
0009   F1FFFE          STBA    charOut,d
0013   FFFD    num:    .WORD   -3
```

Output

```
-3
```

The HEXO system call

Assembly Language Listing

```
0000   C00001          LDWA    DECO,i
0003   390013          SCALL   num,d
0006   D00020          LDBA    ' ',i
0009   F1FFFE          STBA    charOut,d
000C   C00002          LDWA    HEXO,i

0013   FFFD    num:    .WORD   -3
```

Output

```
-3
```

The HEXO system call

Assembly Language Listing

```
0000   C00001          LDWA    DECO,i
0003   390013          SCALL   num,d
0006   D00020          LDBA    ' ',i
0009   F1FFFE          STBA    charOut,d
000C   C00002          LDWA    HEXO,i
000F   390013          SCALL   num,d

0013   FFFD    num:    .WORD   -3
```

Output

```
-3 FFFD
```

The HEXO system call

Assembly Language Listing

```
0000   C00001          LDWA    DECO,i
0003   390013          SCALL   num,d
0006   D00020          LDBA    ' ',i
0009   F1FFFE          STBA    charOut,d
000C   C00002          LDWA    HEXO,i
000F   390013          SCALL   num,d
0012   01              RET
0013   FFFD    num:    .WORD   -3
```

Output

```
-3 FFFD
```

System call macros

System call macros

Operation	Macro	Macro Expansion
Decimal input	`@DECI` OprndSpec,AddrMode	`LDWA DECI,i` `SCALL` OprndSpec,AddrMode
Decimal output	`@DECO` OprndSpec,AddrMode	`LDWA DECO,i` `SCALL` OprndSpec,AddrMode
Hexadecimal output	`@HEXO` OprndSpec,AddrMode	`LDWA DECI,i` `SCALL` OprndSpec,AddrMode
String output	`@STRO` OprndSpec,AddrMode	`LDWA STRO,i` `SCALL` OprndSpec,AddrMode
System call no operation	`@SNOP` OprndSpec,AddrMode	`LDWA SNOP,i` `SCALL` OprndSpec,AddrMode

Assembly Language Source Code

```
            @DECI   width,d     ;Input width
            @DECI   height,d    ;Input height
            LDWA    width,d     ;Compute perimeter of rectangle
            ADDA    height,d
            ASLA
            STWA    perim,d
            @STRO   msg1,d      ;Output "width: "
            @DECO   width,d     ;Output width
            @CHARO  '\n',i      ;Output newline character
            @STRO   msg2,d      ;Output "height: "
            @DECO   height,d    ;Output height
            @CHARO  '\n',i      ;Output newline character
            @STRO   msg3,d      ;Output "perimeter: "
            @DECO   perim,d     ;Output perimeter
            RET
width:      .BLOCK  2
height:     .BLOCK  2
perim:      .BLOCK  2
msg1:       .ASCII  "width: \0"
msg2:       .ASCII  "height: \0"
msg3:       .ASCII  "perimeter: \0"
```

The Assembly Process

The Assembly Process

The von Neumann architecture principles

The Assembly Process

The von Neumann architecture principles

- Main memory stores both instructions and data in binary.

The Assembly Process

The von Neumann architecture principles

- Main memory stores both instructions and data in binary.
- The decode part of the von Neumann cycle interprets the bits fetched from memory into the IR as a CPU instruction.

The Assembly Process

The von Neumann architecture principles

- Main memory stores both instructions and data in binary.
- The decode part of the von Neumann cycle interprets the bits fetched from memory into the IR as a CPU instruction.
- The execute part of the von Neumann cycle interprets the bits processed as data.

The Assembly Process

The von Neumann architecture principles

- Main memory stores both instructions and data in binary.
- The decode part of the von Neumann cycle interprets the bits fetched from memory into the IR as a CPU instruction.
- The execute part of the von Neumann cycle interprets the bits processed as data.
- Main memory does not distinguish instruction bits from data bits.

The Assembly Process

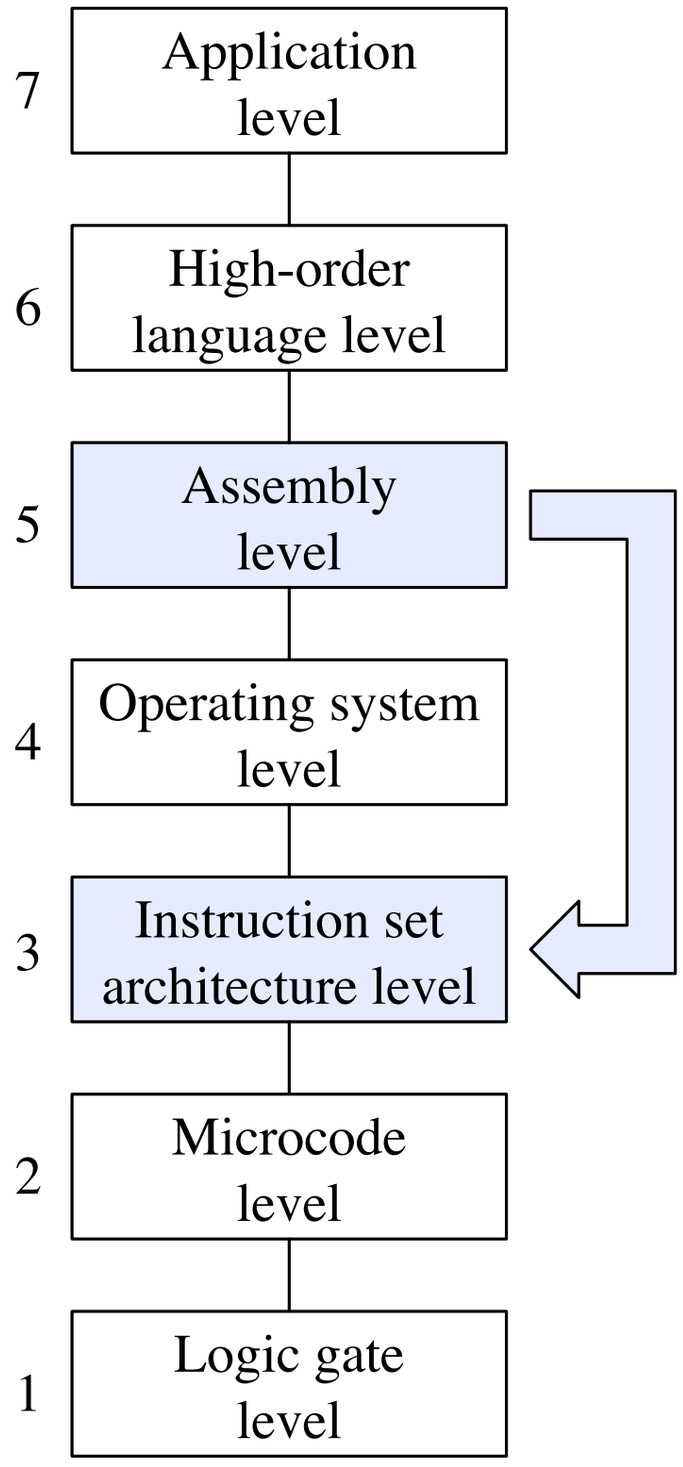
The von Neumann architecture principles

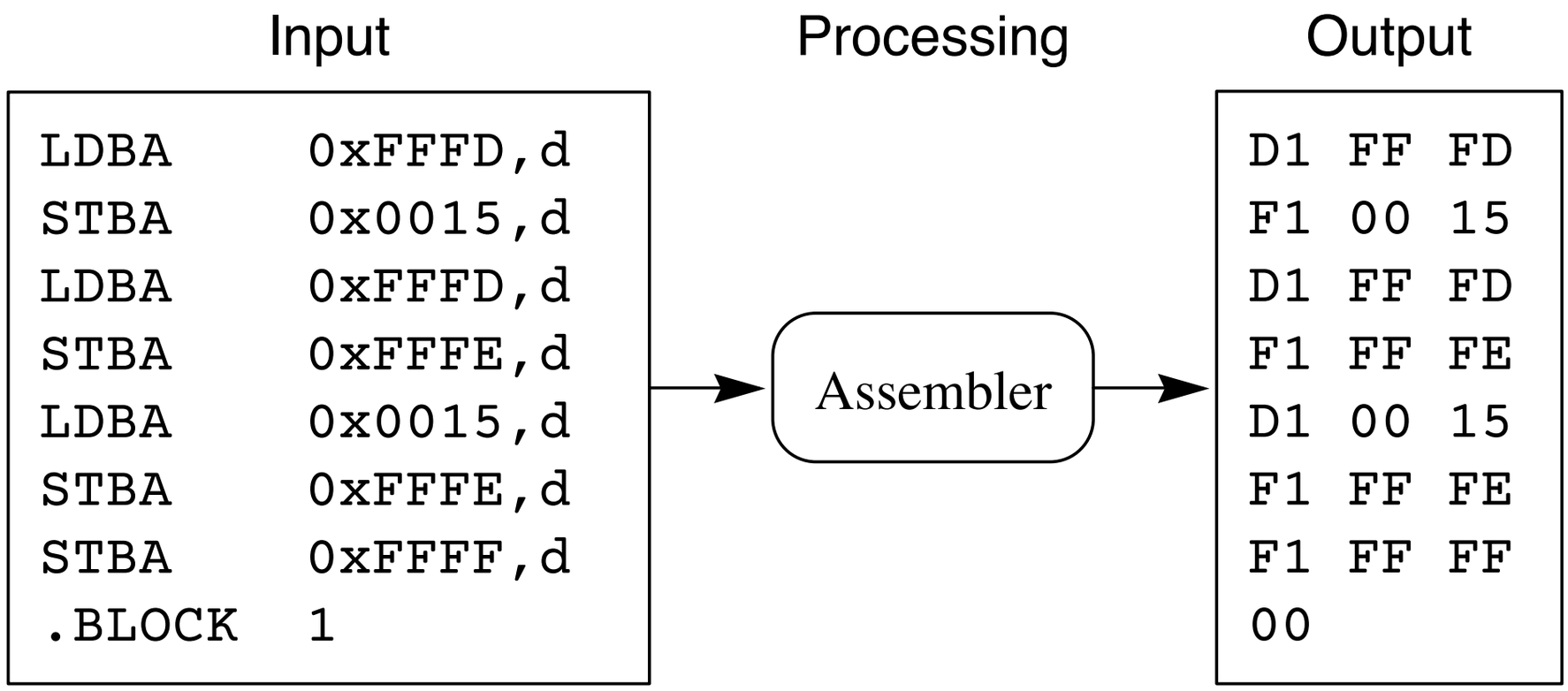
- Main memory stores both instructions and data in binary.
- The decode part of the von Neumann cycle interprets the bits fetched from memory into the IR as a CPU instruction.
- The execute part of the von Neumann cycle interprets the bits processed as data.
- Main memory does not distinguish instruction bits from data bits.
- The CPU can only execute machine language programs. It cannot execute assembly language programs.

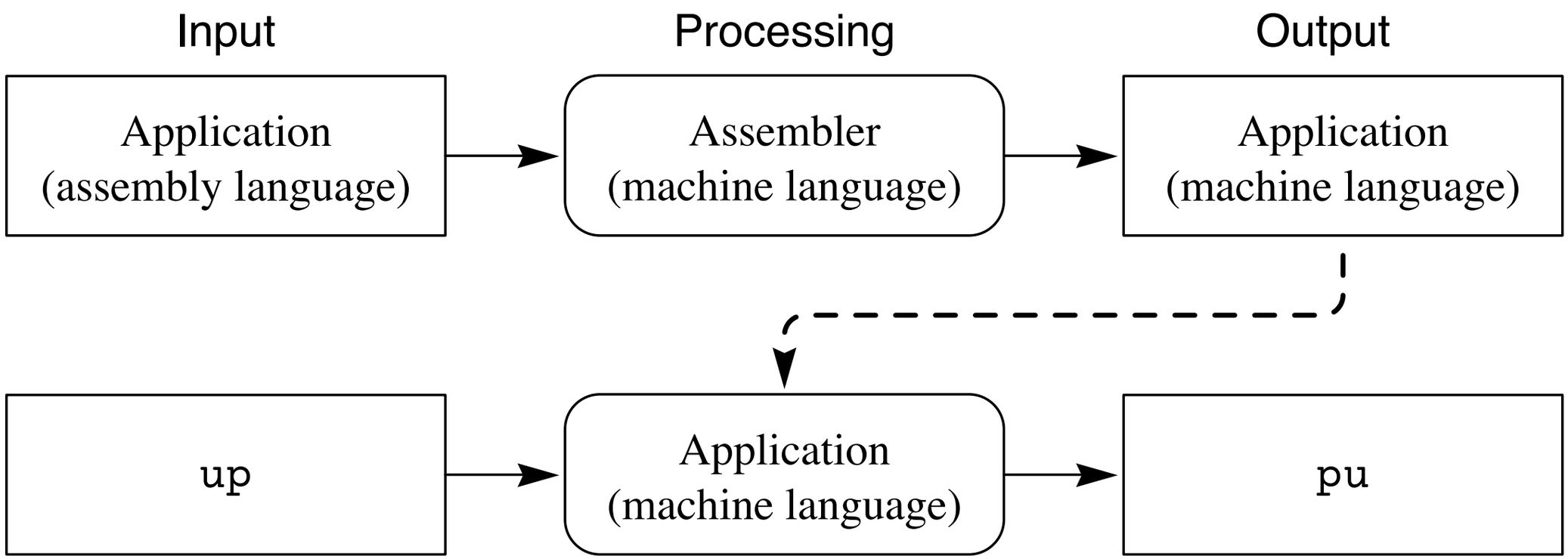
The Assembly Process

The Harvard architecture

- Has separate memories for instructions and data.
- More secure from computer viruses.
- Less flexible than von Neumann architecture.
- Common in calculators and virtual machines.


7
Application level
6
High-order language level
5
Assembly level
4
Operating system level
3
Instruction set architecture level
2
Microcode level
1
Logic gate level


Input
Processing
Output
LDBA 0xFFFD,d
STBA 0x0015,d
LDBA 0xFFFD,d
STBA 0xFFFE,d
LDBA 0x0015,d
STBA 0xFFFE,d
STBA 0xFFFF,d
.BLOCK 1
Assembler
D1 FF FD
F1 00 15
D1 FF FD
F1 FF FE
D1 00 15
F1 FF FE
F1 FF FF
00


Input
Processing
Output
Application
(assembly language)
Assembler
(machine language)
Application
(machine language)
up
Application
(machine language)
pu

Assembly Language Source Code

```
            @DECO   first,d     ;Interpret first as dec
            @CHARO  ' ',i
            @DECO   second,d    ;Interpret second and third as dec
            @CHARO  ' ',i
            @HEXO   second,d    ;Interpret second and third as hex
            @CHARO  ' ',i
            @CHARO  third,d     ;Interpret third as char
            @CHARO  fourth,d    ;Interpret fourth as char
            RET
first:      .WORD   0xFFFE
second:     .BYTE   0x00
third:      .BYTE   'U'
fourth:     .WORD   28673
```

Output

Assembly Language Source Code

```
            @DECO   first,d     ;Interpret first as dec
            @CHARO  ' ',i
            @DECO   second,d    ;Interpret second and third as dec
            @CHARO  ' ',i
            @HEXO   second,d    ;Interpret second and third as hex
            @CHARO  ' ',i
            @CHARO  third,d     ;Interpret third as char
            @CHARO  fourth,d    ;Interpret fourth as char
            RET
first:      .WORD   0xFFFE
second:     .BYTE   0x00
third:      .BYTE   'U'
fourth:     .WORD   28673
```

Output

```
-2
```

Assembly Language Source Code

```
            @DECO   first,d     ;Interpret first as dec
            @CHARO  ' ',i
            @DECO   second,d    ;Interpret second and third as dec
            @CHARO  ' ',i
            @HEXO   second,d    ;Interpret second and third as hex
            @CHARO  ' ',i
            @CHARO  third,d     ;Interpret third as char
            @CHARO  fourth,d    ;Interpret fourth as char
            RET
first:      .WORD   0xFFFE
second:     .BYTE   0x00
third:      .BYTE   'U'
fourth:     .WORD   28673
```

Output

```
-2 85
```

Assembly Language Source Code

```
            @DECO   first,d     ;Interpret first as dec
            @CHARO  ' ',i
            @DECO   second,d    ;Interpret second and third as dec
            @CHARO  ' ',i
            @HEXO   second,d    ;Interpret second and third as hex
            @CHARO  ' ',i
            @CHARO  third,d     ;Interpret third as char
            @CHARO  fourth,d    ;Interpret fourth as char
            RET
first:      .WORD   0xFFFE
second:     .BYTE   0x00
third:      .BYTE   'U'
fourth:     .WORD   28673
```

Output

```
-2 85 0055
```

Assembly Language Source Code

```
          @DECO   first,d     ;Interpret first as dec
          @CHARO  ' ',i
          @DECO   second,d    ;Interpret second and third as dec
          @CHARO  ' ',i
          @HEXO   second,d    ;Interpret second and third as hex
          @CHARO  ' ',i
          @CHARO  third,d     ;Interpret third as char
          @CHARO  fourth,d    ;Interpret fourth as char
          RET
first:    .WORD   0xFFFE
second:   .BYTE   0x00
third:    .BYTE   'U'
fourth:   .WORD   28673
```

Output

```
-2 85 0055 U
```

Assembly Language Source Code

```
            @DECO   first,d     ;Interpret first as dec
            @CHARO  ' ',i
            @DECO   second,d    ;Interpret second and third as dec
            @CHARO  ' ',i
            @HEXO   second,d    ;Interpret second and third as hex
            @CHARO  ' ',i
            @CHARO  third,d     ;Interpret third as char
            @CHARO  fourth,d    ;Interpret fourth as char
            RET
first:      .WORD   0xFFFE
second:     .BYTE   0x00
third:      .BYTE   'U'
fourth:     .WORD   28673
```

Output

```
-2 85 0055 Up
```

Interpreting instruction bits as data

Assembly Language Source Code

```
here:     @DECO    here,d
          RET
```

Output

Interpreting instruction bits as data

Assembly Language Source Code

```
here:     @DECO    here,d
          RET
```

Assembly Language Listing

```
                    ;here:    @DECO    here,d
0000  C00001 here:    LDWA     DECO,i
0003  390000          SCALL    here,d
                    ;         End @DECO
0006  01              RET
```

Output

Interpreting instruction bits as data

Assembly Language Source Code

```
here:     @DECO    here,d
          RET
```

Assembly Language Listing

```
                    ;here:    @DECO    here,d
0000   C00001 here:           LDWA     DECO,i
0003   390000                 SCALL    here,d
                    ;         End @DECO
0006   01                     RET
```

Output

Interpreting instruction bits as data

Assembly Language Source Code

```
here:     @DECO    here,d
          RET
```

Assembly Language Listing

```
                    ;here:    @DECO    here,d
0000   C00001 here:           LDWA     DECO,i
0003   390000                 SCALL    here,d
                    ;         End @DECO
0006   01                     RET
```

Output

Interpreting instruction bits as data

Assembly Language Source Code

```
here:     @DECO    here,d
          RET
```

Assembly Language Listing

```
                    ;here:    @DECO    here,d
0000   C00001 here:           LDWA     DECO,i
0003   390000                 SCALL    here,d
                    ;         End @DECO
0006   01                     RET
```

Output

```
-16384
```

Interpreting data bits as instructions

Assembly Language Source Code

```
            LDWA    num_1,d
            .BYTE   97
            .WORD   0x0012
            STWA    num_3,d
            @DECO   num_3,d
            RET
num_1:      .WORD   47
num_2:      .WORD   15
num_3:      .BLOCK  2
```

Interpreting data bits as instructions

Assembly Language Listing

```
0000   C10010              LDWA     num_1,d
0003   61                  .BYTE    97
0004   0012                .WORD    0x0012
0006   E10014              STWA     num_3,d
                    ;      @DECO    num_3,d
0009   C00001              LDWA     DECO,i
000C   390014              SCALL    num_3,d
                    ;      End @DECO
000F   01                  RET
0010   002F    num_1:      .WORD    47
0012   000F    num_2:      .WORD    15
0014   0000    num_3:      .BLOCK   2
```

Output

Interpreting data bits as instructions

Assembly Language Listing

```
0000   C10010              LDWA     num_1,d
0003   61                  .BYTE    97
0004   0012                .WORD    0x0012
0006   E10014              STWA     num_3,d
                    ;      @DECO    num_3,d
0009   C00001              LDWA     DECO,i
000C   390014              SCALL    num_3,d
                    ;      End @DECO
000F   01                  RET
0010   002F    num_1:      .WORD    47
0012   000F    num_2:      .WORD    15
0014   0000    num_3:      .BLOCK   2
```

Output

97 (dec) = 61 (hex)

Interpreting data bits as instructions

Assembly Language Listing

```
0000   C10010              LDWA     num_1,d
0003   61                  .BYTE    97
0004   0012                .WORD    0x0012
0006   E10014              STWA     num_3,d
                    ;      @DECO    num_3,d
0009   C00001              LDWA     DECO,i
000C   390014              SCALL    num_3,d
                    ;      End @DECO
000F   01                  RET
0010   002F    num_1:      .WORD    47
0012   000F    num_2:      .WORD    15
0014   0000    num_3:      .BLOCK   2
```

Output

Interpreting data bits as instructions

Assembly Language Listing

```
0000   C10010              LDWA     num_1,d
0003   61                  .BYTE    97
0004   0012                .WORD    0x0012
0006   E10014              STWA     num_3,d
                    ;      @DECO    num_3,d
0009   C00001              LDWA     DECO,i
000C   390014              SCALL    num_3,d
                    ;      End @DECO
000F   01                  RET
0010   002F    num_1:      .WORD    47
0012   000F    num_2:      .WORD    15
0014   0000    num_3:      .BLOCK   2
```

Output

61 (hex) = 0110 0001

Interpreting data bits as instructions

Assembly Language Listing

```
0000   C10010              LDWA     num_1,d
0003   61                  .BYTE    97
0004   0012                .WORD    0x0012
0006   E10014              STWA     num_3,d
                    ;      @DECO    num_3,d
0009   C00001              LDWA     DECO,i
000C   390014              SCALL    num_3,d
                    ;      End @DECO
000F   01                  RET
0010   002F    num_1:      .WORD    47
0012   000F    num_2:      .WORD    15
0014   0000    num_3:      .BLOCK   2
```

Output

```
32
```

61 (hex) = 0110 0001

subtract

Interpreting data bits as instructions

Assembly Language Listing

```
0000   C10010              LDWA     num_1,d
0003   61                  .BYTE    97
0004   0012                .WORD    0x0012
0006   E10014              STWA     num_3,d
                  ;        @DECO    num_3,d
0009   C00001              LDWA     DECO,i
000C   390014              SCALL    num_3,d
                  ;        End @DECO
000F   01                  RET
0010   002F    num_1:      .WORD    47
0012   000F    num_2:      .WORD    15
0014   0000    num_3:      .BLOCK   2
```

Output

Interpreting data bits as instructions

Assembly Language Listing

```
0000   C10010              LDWA     num_1,d
0003   61                  .BYTE    97
0004   0012                .WORD    0x0012
0006   E10014              STWA     num_3,d
                    ;      @DECO    num_3,d
0009   C00001              LDWA     DECO,i
000C   390014              SCALL    num_3,d
                    ;      End @DECO
000F   01                  RET
0010   002F    num_1:      .WORD    47
0012   000F    num_2:      .WORD    15
0014   0000    num_3:      .BLOCK   2
```

Output

```
32
```

A program that modifies itself

Assembly Language Source Code

```
            LDWA    num_1,d
there:      SUBA    num_2,d
            STWA    num_3,d
            @DECO   num_3,d
            RET
num_1:      .WORD   243
num_2:      .WORD   7
num_3:      .BLOCK  2
```

A program that modifies itself

Assembly Language Source Code

```
            LDBA    0x81,i
            STBA    there,d
            LDWA    num_1,d
there:      SUBA    num_2,d
            STWA    num_3,d
            @DECO   num_3,d
            RET
num_1:      .WORD   243
num_2:      .WORD   7
num_3:      .BLOCK  2
```

A program that modifies itself

Assembly Language Listing

```
0000   D00081            LDBA    0x0081,i
0003   F10009            STBA    there,d
0006   C10016            LDWA    num_1,d
0009   610018 there:     SUBA    num_2,d
000C   E1001A            STWA    num_3,d
                ;         @DECO   num_3,d
000F   C00001            LDWA    DECO,i
0012   39001A            SCALL   num_3,d
                ;         End @DECO
0015   01                RET
0016   00F3    num_1:    .WORD   243
0018   0007    num_2:    .WORD   7
001A   0000    num_3:    .BLOCK  2
```

Output

```
247
```

A program that modifies itself

Assembly Language Listing

```
0000   D00081              LDBA    0x0081,i
0003   F10009              STBA    there,d
0006   C10016              LDWA    num_1,d
0009   610018  there:      SUBA    num_2,d
000C   E1001A              STWA    num_3,d
                ;          @DECO   num_3,d
000F   C00001              LDWA    DECO,i
0012   39001A              SCALL   num_3,d
                ;          End @DECO
0015   01                  RET
0016   00F3    num_1:      .WORD   243
0018   0007    num_2:      .WORD   7
001A   0000    num_3:      .BLOCK  2
```

81

Output

```
247
```

A program that modifies itself

Assembly Language Listing

```
0000   D00081              LDBA    0x0081,i
0003   F10009              STBA    there,d
0006   C10016              LDWA    num_1,d
0009   610018 there:       SUBA    num_2,d
000C   E1001A              STWA    num_3,d
                ;          @DECO   num_3,d
000F   C00001              LDWA    DECO,i
0012   39001A              SCALL   num_3,d
                ;          End @DECO
0015   01                  RET
0016   00F3    num_1:      .WORD   243
0018   0007    num_2:      .WORD   7
001A   0000    num_3:      .BLOCK  2
```

81

Output

```
247
```

81 (hex) = 1000 0001

A program that modifies itself

Assembly Language Listing

```
0000   D00081            LDBA    0x0081,i
0003   F10009            STBA    there,d
0006   C10016            LDWA    num_1,d
0009   810018  there:    SUBA    num_2,d
000C   E1001A            STWA    num_3,d
                ;         @DECO   num_3,d
000F   C00001             LDWA    DECO,i
0012   39001A             SCALL   num_3,d
                ;         End @DECO
0015   01                RET
0016   00F3    num_1:    .WORD   243
0018   0007    num_2:    .WORD   7
001A   0000    num_3:    .BLOCK  2
```

Output

```
247
```

81

81 (hex) = 1000 0001

OR

A program that modifies itself

Assembly Language Listing

```
0000   D00081            LDBA    0x0081,i
0003   F10009            STBA    there,d
0006   C10016            LDWA    num_1,d
0009   610018 there:     SUBA    num_2,d
000C   E1001A            STWA    num_3,d
                ;         @DECO   num_3,d
000F   C00001             LDWA    DECO,i
0012   39001A             SCALL   num_3,d
                ;         End @DECO
0015   01                RET
0016   00F3    num_1:    .WORD   243
0018   0007    num_2:    .WORD   7
001A   0000    num_3:    .BLOCK  2
```

81

Output

```
247
```

```
          0000 0000 1111 0011
OR        0000 0000 0000 0111
-------------------------------
          0000 0000 1111 0111
```