

APPENDIX

Pep/9 Architecture

This appendix summarizes the architecture of the Pep/9 computer.

FIGURE A.1

The hexadecimal conversion chart.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0_	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1_	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2_	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3_	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4_	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5_	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6_	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7_	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8_	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9_	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A_	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B_	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C_	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D_	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E_	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F_	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

FIGURE A.2

The relationship between hexadecimal and binary.

Hexadecimal	Binary	Hexadecimal	Binary	Hexadecimal	Binary	Hexadecimal	Binary
0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

FIGURE A.3

The American Standard Code for Information Interchange (ASCII).

Char	Bin	Hex									
NUL	000 0000	00	SP	010 0000	20	@	100 0000	40	`	110 0000	60
SOH	000 0001	01	!	010 0001	21	A	100 0001	41	a	110 0001	61
STX	000 0010	02	"	010 0010	22	B	100 0010	42	b	110 0010	62
ETX	000 0011	03	#	010 0011	23	C	100 0011	43	c	110 0011	63
EOT	000 0100	04	\$	010 0100	24	D	100 0100	44	d	110 0100	64
ENQ	000 0101	05	%	010 0101	25	E	100 0101	45	e	110 0101	65
ACK	000 0110	06	&	010 0110	26	F	100 0110	46	f	110 0110	66
BEL	000 0111	07	'	010 0111	27	G	100 0111	47	g	110 0111	67
BS	000 1000	08	(010 1000	28	H	100 1000	48	h	110 1000	68
HT	000 1001	09)	010 1001	29	I	100 1001	49	i	110 1001	69
LF	000 1010	0A	*	010 1010	2A	J	100 1010	4A	j	110 1010	6A
VT	000 1011	0B	+	010 1011	2B	K	100 1011	4B	k	110 1011	6B
FF	000 1100	0C	,	010 1100	2C	L	100 1100	4C	l	110 1100	6C
CR	000 1101	0D	-	010 1101	2D	M	100 1101	4D	m	110 1101	6D
SO	000 1110	0E	.	010 1110	2E	N	100 1110	4E	n	110 1110	6E
SI	000 1111	0F	/	010 1111	2F	O	100 1111	4F	o	110 1111	6F
DLE	001 0000	10	0	011 0000	30	P	101 0000	50	p	111 0000	70
DC1	001 0001	11	1	011 0001	31	Q	101 0001	51	q	111 0001	71
DC2	001 0010	12	2	011 0010	32	R	101 0010	52	r	111 0010	72
DC3	001 0011	13	3	011 0011	33	S	101 0011	53	s	111 0011	73
DC4	001 0100	14	4	011 0100	34	T	101 0100	54	t	111 0100	74
NAK	001 0101	15	5	011 0101	35	U	101 0101	55	u	111 0101	75
SYN	001 0110	16	6	011 0110	36	V	101 0110	56	v	111 0110	76
ETB	001 0111	17	7	011 0111	37	W	101 0111	57	w	111 0111	77
CAN	001 1000	18	8	011 1000	38	X	101 1000	58	x	111 1000	78
EM	001 1001	19	9	011 1001	39	Y	101 1001	59	y	111 1001	79
SUB	001 1010	1A	:	011 1010	3A	Z	101 1010	5A	z	111 1010	7A
ESC	001 1011	1B	;	011 1011	3B	[101 1011	5B	{	111 1011	7B
FS	001 1100	1C	<	011 1100	3C	\	101 1100	5C		111 1100	7C
GS	001 1101	1D	=	011 1101	3D]	101 1101	5D	~	111 1101	7D
RS	001 1110	1E	>	011 1110	3E	^	101 1110	5E	~	111 1110	7E
US	001 1111	1F	?	011 1111	3F	_	101 1111	5F	DEL	111 1111	7F

Abbreviations for Control Characters

NUL	null, or all zeros	FF	form feed	CAN	cancel
SOH	start of heading	CR	carriage return	EM	end of medium
STX	start of text	SO	shift out	SUB	substitute
ETX	end of text	SI	shift in	ESC	escape
EOT	end of transmission	DLE	data link escape	FS	file separator
ENQ	enquiry	DC1	device control 1	GS	group separator
ACK	acknowledge	DC2	device control 2	RS	record separator
BEL	bell	DC3	device control 3	US	unit separator
BS	backspace	DC4	device control 4	SP	space
HT	horizontal tabulation	NAK	negative acknowledge	DEL	delete
LF	line feed	SYN	synchronous idle		
VT	vertical tabulation	ETB	end of transmission block		

FIGURE A.4

The central processing unit of the Pep/9 computer.

Central processing unit (CPU)

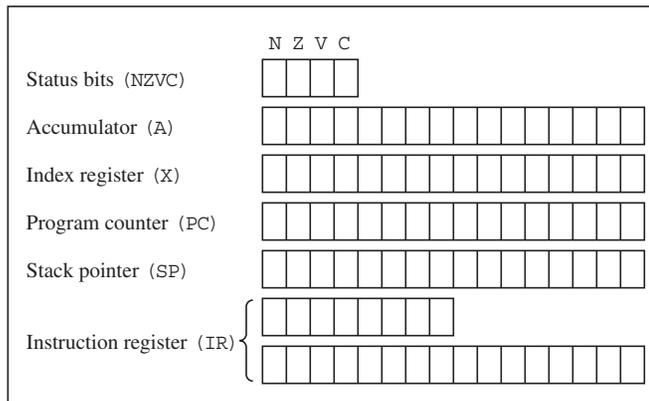
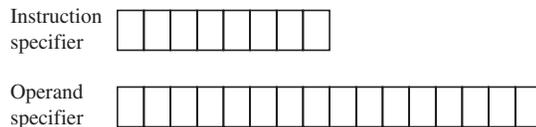
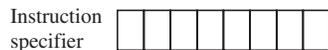


FIGURE A.5

The Pep/9 instruction format.



(a) The two parts of a nonunary instruction.



(b) A unary instruction.

FIGURE A.6

The Pep/9 instruction specifier fields.

aaa	Addressing Mode	a	Addressing Mode	r	Register
000	Immediate	0	Immediate	0	Accumulator, A
001	Direct	1	Indexed	1	Index register, X
010	Indirect	(b) The addressing-a field.		(c) The register-r field.	
011	Stack-relative				
100	Stack-relative deferred				
101	Indexed				
110	Stack-indexed				
111	Stack-deferred indexed				

(a) The addressing-aaa field.

FIGURE A.7

The Pep/9 addressing modes.

Addressing Mode	aaa	Letters	Operand
Immediate	000	i	OprndSpec
Direct	001	d	Mem[OprndSpec]
Indirect	010	n	Mem[Mem[OprndSpec]]
Stack-relative	011	s	Mem[SP + OprndSpec]
Stack-relative deferred	100	sf	Mem[Mem[SP + OprndSpec]]
Indexed	101	x	Mem[OprndSpec + X]
Stack-indexed	110	sx	Mem[SP + OprndSpec + X]
Stack-deferred indexed	111	sfx	Mem[Mem[SP + OprndSpec] + X]

FIGURE A.8

The Pep/9 instruction set at Level Asmb5.

Instruction Specifier	Mnemonic	Instruction	Addressing Mode	Status Bits
0000 0000	STOP	Stop execution	U	
0000 0001	RET	Return from CALL	U	
0000 0010	RETTR	Return from trap	U	
0000 0011	MOVSPA	Move SP to A	U	
0000 0100	MOVFLGA	Move NZVC flags to A(12..15)	U	
0000 0101	MOVAFLG	Move A(12..15) to NZVC flags	U	
0000 011r	NOTr	Bitwise invert r	U	NZ
0000 100r	NEGr	Negate r	U	NZV
0000 101r	ASLr	Arithmetic shift left r	U	NZVC
0000 110r	ASRr	Arithmetic shift right r	U	NZC
0000 111r	ROLr	Rotate left r	U	C
0001 000r	RORr	Rotate right r	U	C
0001 001a	BR	Branch unconditional	i, x	
0001 010a	BRLE	Branch if less than or equal to	i, x	
0001 011a	BRLT	Branch if less than	i, x	
0001 100a	BREQ	Branch if equal to	i, x	
0001 101a	BRNE	Branch if not equal to	i, x	
0001 110a	BRGE	Branch if greater than or equal to	i, x	
0001 111a	BRGT	Branch if greater than	i, x	
0010 000a	BRV	Branch if V	i, x	
0010 001a	BRC	Branch if C	i, x	
0010 010a	CALL	Call subroutine	i, x	
0010 011n	NOPn	Unary no operation trap	U	
0010 1aaa	NOP	Nonunary no operation trap	i	

(continued)

FIGURE A.8The Pep/9 instruction set at Level Asmb5. (*continued*)

0011 0aaa	DECI	Decimal input trap	d, n, s, sf, x, sx, sfx	NZV
0011 1aaa	DECO	Decimal output trap	i, d, n, s, sf, x, sx, sfx	
0100 0aaa	HEXO	Hexadecimal output trap	i, d, n, s, sf, x, sx, sfx	
0100 1aaa	STRO	String output trap	d, n, s, sf, x	
0101 0aaa	ADDSP	Add to stack pointer (SP)	i, d, n, s, sf, x, sx, sfx	NZVC
0101 1aaa	SUBSP	Subtract from stack pointer (SP)	i, d, n, s, sf, x, sx, sfx	NZVC
0110 raaa	ADDr	Add to r	i, d, n, s, sf, x, sx, sfx	NZVC
0111 raaa	SUBr	Subtract from r	i, d, n, s, sf, x, sx, sfx	NZVC
1000 raaa	ANDr	Bitwise AND to r	i, d, n, s, sf, x, sx, sfx	NZ
1001 raaa	ORr	Bitwise OR 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	

FIGURE A.9

The pseudo-ops of Pep/9 assembly language.

Pseudo-op	Assembler Directive
.ADDRSS	The address of a symbol
.ALIGN	Padding to align at a memory boundary
.ASCII	A string of ASCII bytes
.BLOCK	A block of zero bytes
.BURN	Initiate ROM burn
.BYTE	A byte value
.END	The sentinel for the assembler
.EQUATE	Equate a symbol to a constant value
.WORD	A word value

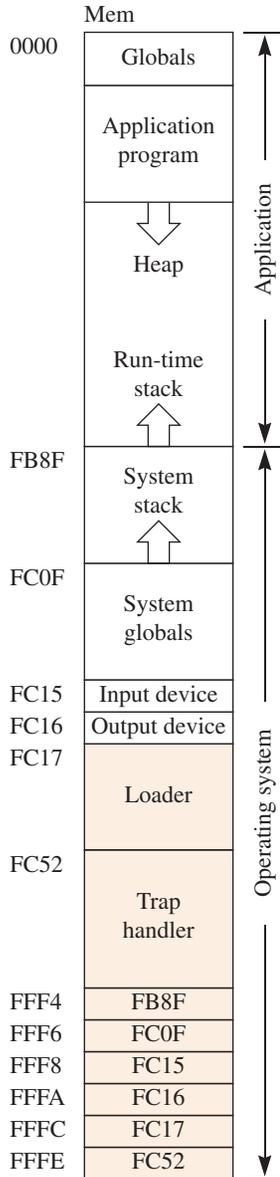


FIGURE A.10

A memory map of the Pep/9 memory. The shaded part is read-only memory.

FIGURE A.11

The RTL specification of Pep/9 instructions.

Instruction	Register Transfer Language Specification
STOP	Stop execution
RET	$PC \leftarrow \text{Mem}[\text{SP}]; \text{SP} \leftarrow \text{SP} + 2$
RETRR	$\text{NZVC} \leftarrow \text{Mem}[\text{SP}\langle 4..7 \rangle]; A \leftarrow \text{Mem}[\text{SP} + 1]; X \leftarrow \text{Mem}[\text{SP} + 3]; PC \leftarrow \text{Mem}[\text{SP} + 5]; \text{SP} \leftarrow \text{Mem}[\text{SP} + 7]$
MOVSPA	$A \leftarrow \text{SP}$
MOVFLGA	$A\langle 8..11 \rangle \leftarrow 0, A\langle 12..15 \rangle \leftarrow \text{NZVC}$
MOVAFLG	$\text{NZVC} \leftarrow A\langle 12..15 \rangle$
NOTr	$r \leftarrow \neg r; N \leftarrow r < 0, Z \leftarrow r = 0$
NEGr	$r \leftarrow \neg r; N \leftarrow r < 0, Z \leftarrow r = 0, V \leftarrow \{\text{overflow}\}$
ASLr	$C \leftarrow r\langle 0 \rangle, r\langle 0..14 \rangle \leftarrow r\langle 1..15 \rangle, r\langle 15 \rangle \leftarrow 0; N \leftarrow r < 0, Z \leftarrow r = 0, V \leftarrow \{\text{overflow}\}$
ASRr	$C \leftarrow r\langle 15 \rangle, r\langle 1..15 \rangle \leftarrow r\langle 0..14 \rangle; N \leftarrow r < 0, Z \leftarrow r = 0$
ROLr	$C \leftarrow r\langle 0 \rangle, r\langle 0..14 \rangle \leftarrow r\langle 1..15 \rangle, r\langle 15 \rangle \leftarrow C$
RORr	$C \leftarrow r\langle 15 \rangle, r\langle 1..15 \rangle \leftarrow r\langle 0..14 \rangle, r\langle 0 \rangle \leftarrow C$
BR	$PC \leftarrow \text{Oprnd}$
BRLE	$N = 1 \vee Z = 1 \Rightarrow PC \leftarrow \text{Oprnd}$
BRLT	$N = 1 \Rightarrow PC \leftarrow \text{Oprnd}$
BREQ	$Z = 1 \Rightarrow PC \leftarrow \text{Oprnd}$
BRNE	$Z = 0 \Rightarrow PC \leftarrow \text{Oprnd}$
BRGE	$N = 0 \Rightarrow PC \leftarrow \text{Oprnd}$
BRGT	$N = 0 \wedge Z = 0 \Rightarrow PC \leftarrow \text{Oprnd}$
BRV	$V = 1 \Rightarrow PC \leftarrow \text{Oprnd}$
BRC	$C = 1 \Rightarrow PC \leftarrow \text{Oprnd}$
CALL	$\text{SP} \leftarrow \text{SP} - 2; \text{Mem}[\text{SP}] \leftarrow PC; PC \leftarrow \text{Oprnd}$
NOPn	Trap: Unary no operation
NOP	Trap: Nonunary no operation
DECI	Trap: $\text{Oprnd} \leftarrow \{\text{decimal input}\}$
DECO	Trap: $\{\text{decimal output}\} \leftarrow \text{Oprnd}$
HEXO	Trap: $\{\text{hexadecimal output}\} \leftarrow \text{Oprnd}$
STRO	Trap: $\{\text{string output}\} \leftarrow \text{Oprnd}$
ADDSP	$\text{SP} \leftarrow \text{SP} + \text{Oprnd}$
SUBSP	$\text{SP} \leftarrow \text{SP} - \text{Oprnd}$
ADDR	$r \leftarrow r + \text{Oprnd}; N \leftarrow r < 0, Z \leftarrow r = 0, V \leftarrow \{\text{overflow}\}, C \leftarrow \{\text{carry}\}$
SUBr	$r \leftarrow r - \text{Oprnd}; N \leftarrow r < 0, Z \leftarrow r = 0, V \leftarrow \{\text{overflow}\}, C \leftarrow \{\text{carry}\}$

ANDr	$r \leftarrow r \wedge \text{Oprnd}; N \leftarrow r < 0, Z \leftarrow r = 0$
ORr	$r \leftarrow r \vee \text{Oprnd}; N \leftarrow r < 0, Z \leftarrow r = 0$
CPWr	$T \leftarrow r - \text{Oprnd}; N \leftarrow T < 0, Z \leftarrow T = 0, V \leftarrow \{\text{overflow}\}, C \leftarrow \{\text{carry}\}; N \leftarrow N \oplus V$
CPBr	$T \leftarrow r(8..15) - \text{byte Oprnd}; N \leftarrow T < 0, Z \leftarrow T = 0, V \leftarrow 0, C \leftarrow 0$
LDWr	$r \leftarrow \text{Oprnd}; N \leftarrow r < 0, Z \leftarrow r = 0$
LDBr	$r(8..15) \leftarrow \text{byte Oprnd}; N \leftarrow 0, Z \leftarrow r(8..15) = 0$
STWr	$\text{Oprnd} \leftarrow r$
STBr	$\text{byte Oprnd} \leftarrow r(8..15)$
Trap	$T \leftarrow \text{Mem}[\text{FFF6}]; \text{Mem}[T - 1] \leftarrow \text{IR}(0..7); \text{Mem}[T - 3] \leftarrow \text{SP}; \text{Mem}[T - 5] \leftarrow \text{PC};$ $\text{Mem}[T - 7] \leftarrow X; \text{Mem}[T - 9] \leftarrow A; \text{Mem}[T - 10](4..7) \leftarrow \text{NZVC}; \text{SP} \leftarrow T - 10;$ $\text{PC} \leftarrow \text{Mem}[\text{FFFE}]$

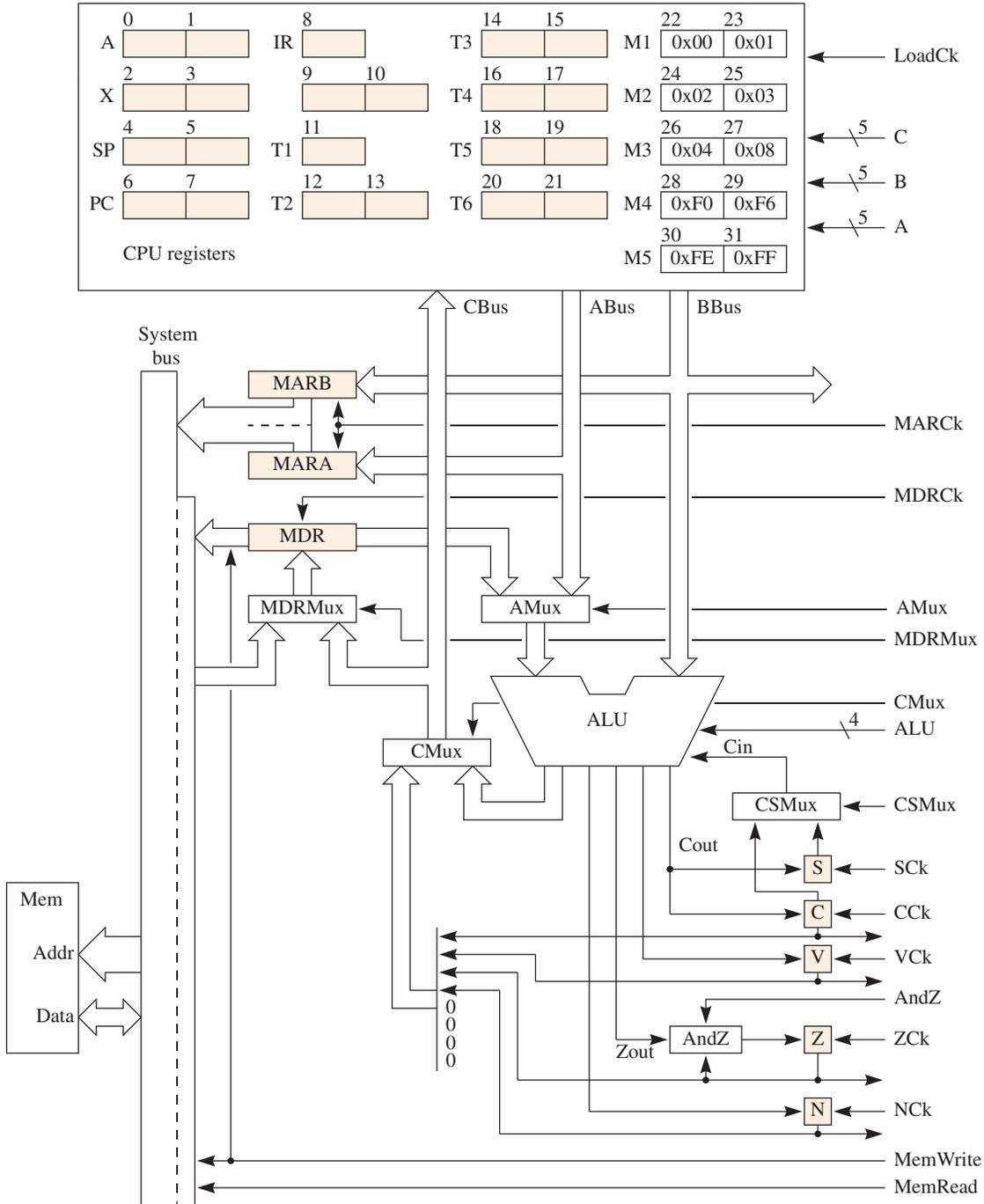
FIGURE A.12

The 16 functions of the Pep/9 ALU.

ALU Control		Result	Status Bits			
(bin)	(dec)		N	Zout	V	Cout
0000	0	A	N	Z	0	0
0001	1	A plus B	N	Z	V	C
0010	2	A plus B plus Cin	N	Z	V	C
0011	3	A plus \bar{B} plus 1	N	Z	V	C
0100	4	A plus \bar{B} plus Cin	N	Z	V	C
0101	5	$A \cdot B$	N	Z	0	0
0110	6	$\overline{A \cdot B}$	N	Z	0	0
0111	7	A + B	N	Z	0	0
1000	8	$\overline{A + B}$	N	Z	0	0
1001	9	$A \oplus B$	N	Z	0	0
1010	10	\bar{A}	N	Z	0	0
1011	11	ASL A	N	Z	V	C
1100	12	ROL A	N	Z	V	C
1101	13	ASR A	N	Z	0	C
1110	14	ROR A	N	Z	0	C
1111	15	0	A<4>	A<5>	A<6>	A<7>

FIGURE A.13

The data section of the Pep/9 CPU.

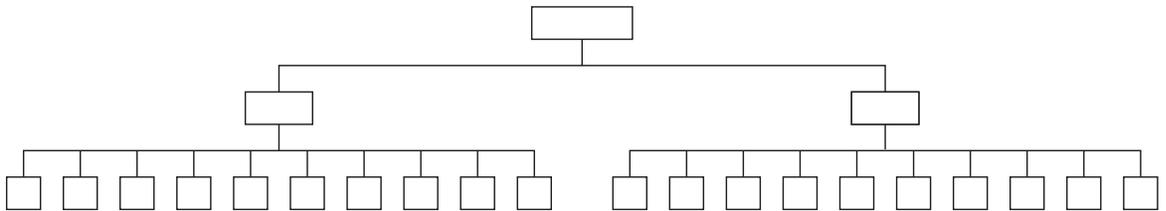


Solutions to Selected Exercises

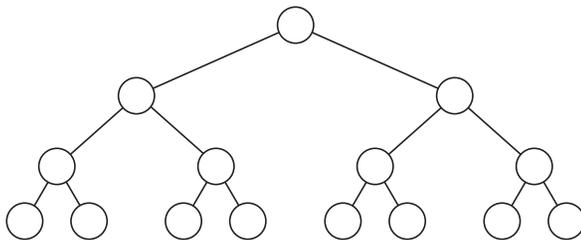
Chapter 1

2. (a) 11,110 not counting Khan

3. (a)



4. (a)



(b) 31

9. 43 ms

12. 32 b/s

15. (a) 1936 b (b) 152 characters

21. Temp5

F.Name	F.Major	F.State
Ron	Math	OR

Temp6

S.Name	S.Class	S.Major	S.State
Beth	Soph	Hist	TX
Allison	Soph	Math	AZ

22. (a) select Sor where S.Name = Beth giving Temp
project Temp over S.State giving Result

Chapter 2

1. (a) Called four times.

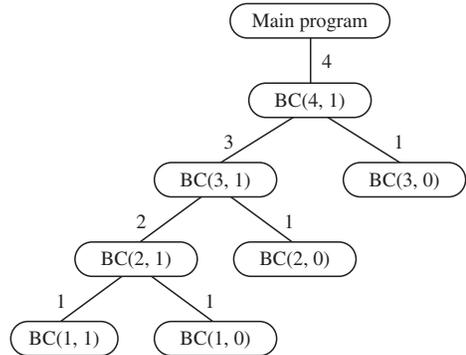
2.

(a, part 1)

```

Main program
  Call BC(4, 1)
    Call BC(3, 1)
      Call BC(2, 1)
        Call BC(1, 1)
          Return to BC(2, 1)
            Call BC(1, 0)
              Return to BC(2, 1)
                Return to BC(3, 1)
                  Call BC(2, 0)
                    Return to BC(3, 1)
                      Return to BC(4, 1)
                        Call BC(3, 0)
                          Return to BC(4, 1)
                            Return to main program
  
```

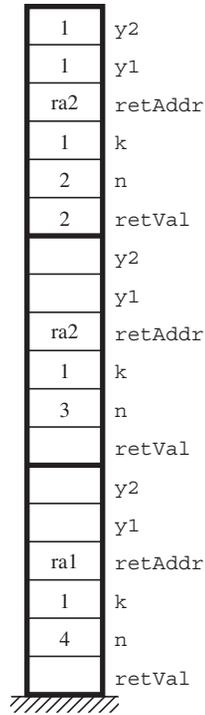
(a, part 2)



(a, part 3) Called seven times.

(a, part 4) Maximum of five stack frames.

(a, part 5)

**Chapter 3**

1. (a) Octal: 267, 270, 271, 272, 273, 274, 275, 276, 277, 300, 301
 (b) Base 3: 2102, 2110, 2111, 2112, 2120, 2121, 2122, 2200, 2201, 2202, 2210
 (c) Binary: 10101, 10110, 10111, 11000, 11001, 11010, 11011, 11100, 11101, 11110, 11111
 (d) Base 5: 2433, 2434, 2440, 2441, 2442, 2443, 2444, 3000, 3001, 3002
3. (a) 18 (b) 6 (c) 11 (d) 8 (e) 31 (f) 85
5. (a) 11001 (b) 10000 (c) 1 (d) 1110 (e) 101 (f) 101001
7. (a) 00 to 11 (bin) and 0 to 3 (dec) (b) 000 to 111 (bin) and 0 to 7 (dec)
8. (a) 111 0100, C = 0 (b) 001 0000, C = 1
 (c) 111 1110, C = 1 (d) 000 0000, C = 0
11. (a) $7 \times 8^4 + 0 \times 8^3 + 1 \times 8^2 + 4 \times 8^1 + 6 \times 8^0$
13. (a) 011 0001 (b) 110 0101 (c) 000 0000 (d) 100 0000 (e) 111 1111
 (f) 111 1110 (g) 100 0000 to 011 1111 (bin) and -64 to 63 (dec)
15. (a) 29 (b) -43 (c) -4 (d) 1 (e) -64 (f) -63

- 17.** (a) 011 1001, NZVC = 0000 (b) 000 0110, NZVC = 0001
 (c) 001 1011, NZVC = 0011 (d) 101 0110, NZVC = 1001
 (e) 100 0001, NZVC = 1011 (f) 111 0100, NZVC = 1000
- 19.** (a) 10 to 01 (bin) and -2 to 1 (dec)
 (b) 100 to 011 (bin) and -4 to 3 (dec)
- 20.** (a) 010 1000, NZ = 00 (b) 000 0101, NZ = 00 (c) 110 1110, NZ = 10
 (d) 101 1111, NZ = 10 (e) 100 0110, NZ = 10 (f) 101 1010, NZ = 10
 (g) 101 0100 (h) 001 0101
- 22.** (a) 24 (dec) = 001 1000 (bin)
 ASL 001 1000 = 011 0000 (bin) = 48 (dec), NZVC = 0000
 ASR 001 1000 = 000 1100 (bin) = 12 (dec), NZC = 000
 (b) 37 (dec) = 010 0101 (bin)
 ASL 010 0101 = 100 1010 (bin) = -54 (dec), NZVC = 1010
 ASR 010 0101 = 001 0010 (bin) = 18 (dec), NZC = 001
 (c) -26 (dec) = 110 0110 (bin)
 ASL 110 0110 = 100 1100 (bin) = -52 (dec), NZVC = 1001
 ASR 110 0110 = 111 0011 (bin) = -13 (dec), NZC = 100
 (d) 1 (dec) = 000 0001 (bin)
 ASL 000 0001 = 000 0010 (bin) = 2 (dec), NZVC = 0000
 ASR 000 0001 = 000 0000 (bin) = 0 (dec), NZC = 011
 (e) 0 (dec) = 000 0000 (bin)
 ASL 000 0000 = 000 0000 (bin) = 0 (dec), NZVC = 0100
 ASR 000 0000 = 000 0000 (bin) = 0 (dec), NZC = 010
 (f) -1 (dec) = 111 1111 (bin)
 ASL 111 1111 = 111 1110 (bin) = -2 (dec), NZVC = 1001
 ASR 111 1111 = 111 1111 (bin) = -1 (dec), NZC = 101
- 25.** (a) C = 1, ROL 010 1101 = 101 1011, C = 0
 (b) C = 0, ROL 010 1101 = 101 1010, C = 0
 (c) C = 1, ROR 010 1101 = 101 0110, C = 1
 (d) C = 0, ROR 010 1101 = 001 0110, C = 1
- 28.** (a) 3AB7, 3AB8, 3AB9, 3ABA, 3ABB, 3ABC
- 29.** (a) 11,614
- 30.** (a) 68CF
- 32.** (a) 5D (hex) = 101 1101 (bin) = -35 (dec)
 (b) 2F (hex) = 010 1111 (bin) = 47 (dec)
 (c) 40 (hex) = 100 0000 (bin) = -256 (dec)
- 34.** (a) -27 (dec) = 110 0101 (bin) = 65 (hex)
 (b) 63 (dec) = 011 1111 (bin) = 3F (hex)
 (c) -1 (dec) = 111 1111 (bin) = 7F (hex)
- 36.** Have a nice day!

38. 101 0000 110 0001 111 1001 010 0000 010 0100
 011 0000 010 1110 011 1001 011 0010

40. (a) D5 82

43. (a) An octal digit represents three bits.

44. (a) 6.640625 (b) 0.046875 (c) 1.0

46. (a) 1101.00101 (b) 0.000101 (c) 0.1001100110011...

50. (a) -12.5 (dec) = -1100.1 (bin), which is stored as 1 110 1001

51. (a) 0.90625

53. (a) 1.0×2^6

Chapter 4

1. (a) 65,536 bytes (b) 32,768 words (c) 524,288 bits
 (d) 92 bits (e) 5699 times bigger

3. For instruction 6AF82C For instruction D623D0
 (a) opcode = 0110 (a) opcode = 1101
 (b) It adds to register r (b) It loads a byte from memory to register r
 (c) r = 1 (c) r = 0
 (d) The index register, X (d) The accumulator, A
 (e) aaa = 010 (e) aaa = 110
 (f) Indirect (f) Stack-indexed
 (g) OprndSpec = F82C (g) OprndSpec = 23D0

5.

	A	X	Mem[0A3F]	Mem[0A41]
Original content	19AC	FE20	FF00	103D
(a) C1 = Load word accumulator	<u>FF00</u>	FE20	FF00	103D
(b) D1 = Load byte accumulator	<u>19FF</u>	FE20	FF00	103D
(c) D9 = Load byte index register	19AC	<u>FE10</u>	FF00	103D
(d) F1 = Store byte accumulator	19AC	FE20	FF00	<u>AC3D</u>
(e) E9 = Store word index register	19AC	FE20	<u>FE20</u>	103D
(f) 79 = Subtract index register	19AC	<u>EDE3</u>	FF00	103D
(g) 71 = Subtract accumulator	<u>1AAC</u>	FE20	FF00	103D
(h) 91 = OR accumulator	<u>FFAC</u>	FE20	FF00	103D
(i) 07 = Invert index register	19AC	<u>01DF</u>	FF00	103D

9. (a) M

Chapter 5

1. (a) ORX 0xEF2A, n (b) MOVSPA (c) LDBA 0x003D, sfX

3. (a) 0A (b) 33 00 0F (c) 1A 01 E6

5. (a) 42 65 61 72 00 (b) F8 (c) 03 16

7. mug
10. -57
72
0048
Hi
12. (a) Object code is:
- ```
38 00 6D D0 00 0A F1 FC 16 38 6D 6D D0 00 0A F1
FC 16 D0 00 26 F1 FC 16 00 zz
```

Output is:

```
109
28013
&
```

13. Object code is:
- ```
12 00 05 00 09 39 00 03 00 zz
```
- Symbol `here` has value 0003 (hex). Symbol `there` has value 0005 (hex).
15. Symbol `this` has value 0000 (hex). The output is 4100. The output comes from the hexadecimal output instruction, which outputs its own instruction specifier followed by the first byte of its operand specifier.
18. The compiler uses its symbol table to store the type of each variable. It consults the symbol table whenever it encounters an expression or assignment statement to verify that the types are compatible.

Chapter 6

3. Because the current value of `j` will be in the accumulator regardless of whether control comes from the `STWA` at 0009 or from the `BR` at the bottom of the loop. Before the `BR` at the bottom of the loop, the accumulator was used to increment `j`; hence, its current value will still be in the accumulator when `CPWA` executes.

6.



(a) Before execution of the second `CALL`.

(b) After execution of the second `CALL`.

8. The branch address is calculated as

$$\begin{aligned}
 \text{Oprnd} &= \text{Mem}[\text{OprndSpec} + X] \\
 &= \text{Mem}[0013 + 8] \\
 &= \text{Mem}[001B] \\
 &= 4900
 \end{aligned}$$

You cannot tell from the program listing what bits are at 4900, but assuming that they are all 0's, the von Neumann cycle blindly interprets the 00 at address 4900 as the STOP instruction.

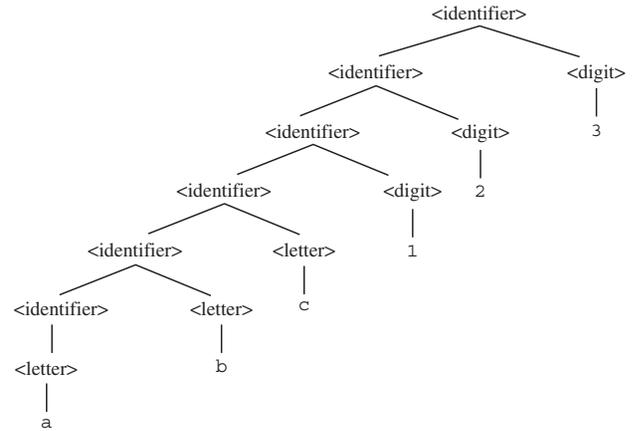
Chapter 7

1. The fundamental question of computer science is: What can be automated?

3. (a)

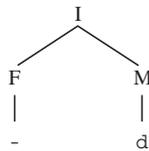
<identifier>

\Rightarrow <identifier> <digit>	Rule 3
\Rightarrow <identifier> 3	Rule 9
\Rightarrow <identifier> <digit> 3	Rule 3
\Rightarrow <identifier> 23	Rule 8
\Rightarrow <identifier> <digit> 23	Rule 3
\Rightarrow <identifier> 123	Rule 7
\Rightarrow <identifier> <letter> 123	Rule 2
\Rightarrow <identifier> c123	Rule 6
\Rightarrow <identifier> <letter> c123	Rule 2
\Rightarrow <identifier> bc123	Rule 5
\Rightarrow <letter> bc123	Rule 1
\Rightarrow abc123	Rule 4



4. (a)

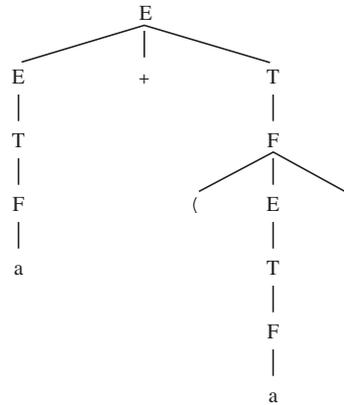
I \Rightarrow FM	Rule 1
\Rightarrow -M	Rule 3
\Rightarrow -d	Rule 6



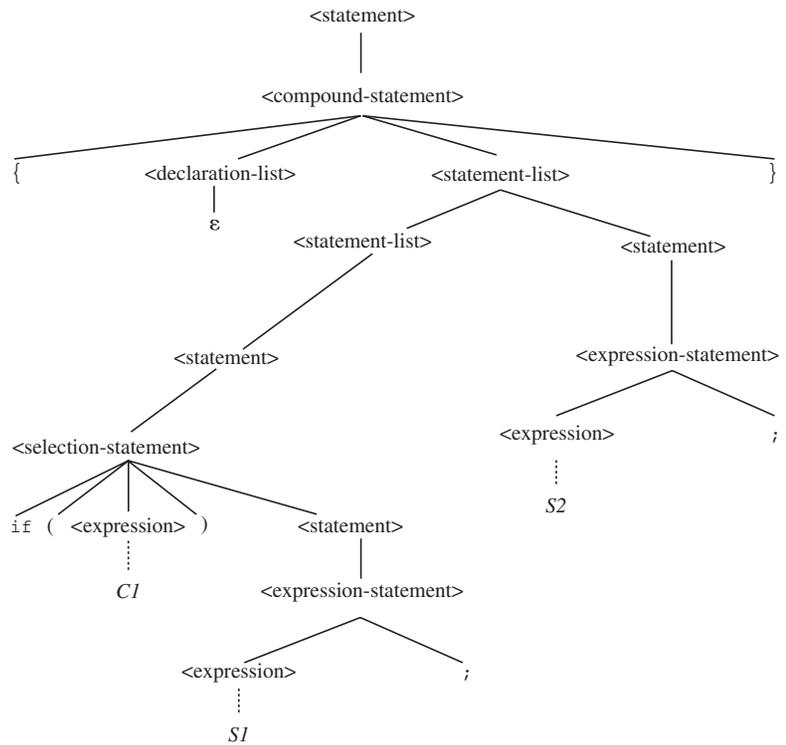
5. (a)

A \Rightarrow abC	Rule 2
\Rightarrow abc	Rule 5

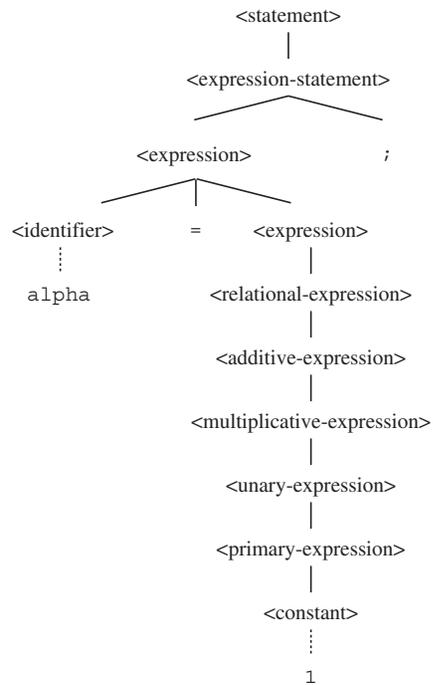
6. (a)



7. (a)

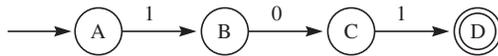


8. (a)



11. (a) The machine is deterministic. There are no inaccessible states.

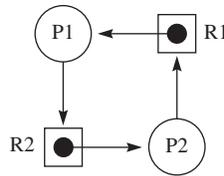
13. (a)

**Chapter 8**

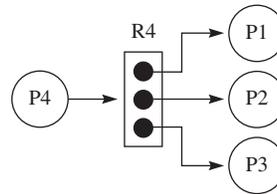
2. (a) 36 (hex), the ASCII code for 6 in the 30th byte, 6C
 (b) 36 (hex)
 (c) 60 (hex)
4. (a) 31 (hex), the instruction specifier of the interrupted instruction
 (b) 0006 (hex)
5. (a) 39 (hex) the instruction specifier of the interrupted instruction
 (b) 0007
6. (a) 49 (hex) the instruction specifier of the interrupted instruction
 (b) 0009
7. (a) 0003 (hex), the numeric value of the ASCII character 3
 (b) 0007 (hex), the numeric value of the ASCII character 7
 (c) 0000 (hex), the value of the `init` state
8. Hint: The first character input is the ASCII hyphen character

9. (a) 0025 (hex), which is 37 (dec) (b) 0025 (hex), not negated because it was already nonnegative (c) The `CALL` from FF78 is for writing the 100's place. The accumulator still contains 0025 (hex), which is 37 (dec), because $37 \bmod 100$ is 37.
18. (a) The algorithm no longer guarantees mutual exclusion. Suppose P1 and P2 are both in their remainder sections with `enter1` and `enter2` both false. P1 could execute its `while` loop test and be interrupted, after which P2 could execute its `while` loop test. They could then assign their respective `enter` variables to true and enter their critical sections simultaneously.
20. (a) $s = 0$ and there are no blocked processes.
22. (a) The algorithm does guarantee mutual exclusion. If you remove the τ semaphore altogether, you get the algorithm of Figure 8.20, which guarantees mutual exclusion regardless of any other code present in the algorithm.
24. (a) Mutual exclusion is no longer guaranteed. Can you find a trace that allows P1 and P2 to enter their critical sections simultaneously? Deadlock, however, cannot occur.

25.



(a) Contains a deadlock cycle.



(b) Does not contain a cycle and therefore does not have a deadlock.

Chapter 9

2. (a) A single bound register would suffice because only one process at a time can execute. If a user process tries to access a memory location outside its logical address space, the hardware must interrupt the access before the page table because there is no such page in main memory. The operating system must keep track of one bound value for each process.
4. (a) 2^{12} or 4096 bytes
6. Those that have a dirty bit value of N—namely, the ones in frames 2, 5, and 6.

8. The start of one page reference sequence for a job that has three frames allocated is 1, 2, 3, 1, 4, 2, . . . , which has four faults for FIFO and five for LRU. Can you complete the sequence in such a way that FIFO is better in this special case?
10. It produces five faults compared to seven for FIFO and six for LRU. You should trace the algorithm to verify this figure.
11. (a) Hint: The worst case occurs when the start of the block has just rotated past the read/write head when the head has reached the track. Hence, it is time for the disk to make one complete revolution. You can calculate it from the RPM number.
12. (a) Four data bits. (b) One parity bit.
16. (a) An error did occur in position 2. The corrected code word is 1101 1010 1001.

Chapter 10

1. (a)

$$\begin{aligned}
 & x + 1 \\
 = & \langle \text{complement of OR} \rangle \\
 & x + (x + x') \\
 = & \langle \text{associative of OR} \rangle \\
 & (x + x) + x' \\
 = & \langle \text{idempotent of OR} \rangle \\
 & x + x' \\
 = & \langle \text{complement of OR} \rangle \\
 & 1
 \end{aligned}$$

4. To show that the complement of $a + b$ is $a' \cdot b'$, you must show that $(a + b) \cdot (a' \cdot b') = 0$ and $(a + b) + (a' \cdot b') = 1$

First part:

$$\begin{aligned}
 & (a + b) \cdot (a' \cdot b') \\
 = & \langle \text{commutative of AND} \rangle \\
 & (a' \cdot b') \cdot (a + b) \\
 = & \langle \text{distributive of AND over OR} \rangle \\
 & (a' \cdot b') \cdot a + (a' \cdot b') \cdot b \\
 = & \langle \text{commutative and associative of AND} \rangle \\
 & b' \cdot (a \cdot a') + a' \cdot (b \cdot b') \\
 = & \langle \text{complement of AND} \rangle \\
 & b' \cdot 0 + a' \cdot 0
 \end{aligned}$$

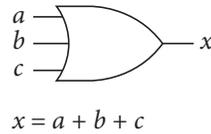
$$= \langle \text{the zero theorem of AND, } x \cdot 0 = 0 \rangle$$

$$0 + 0$$

$$= \langle \text{idempotent of OR} \rangle$$

$$0$$

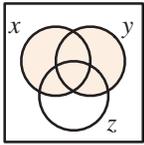
8. (a)



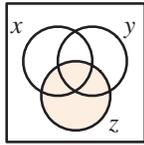
a	b	c	x
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

9. (a) The union of any set with the empty set is the set itself.

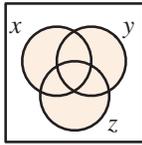
10. (a)



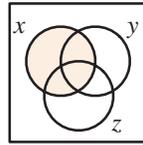
(1) $(x + y)$



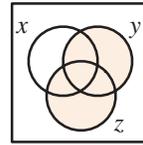
(2) z



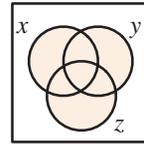
(3) $(x + y) + z$



(4) x



(5) $y + z$

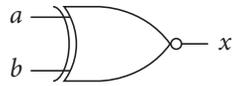


(6) $x + (y + z)$

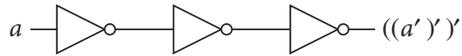
13. (a)

x	y	$x \text{ AND } y$
0	0	0
0	A	0
0	B	0
0	0	0
A	1	0
A	0	A
A	A	0
A	B	A
B	1	0
B	0	0
B	A	B
B	B	B
1	1	0
1	A	A
1	B	B
1	1	1

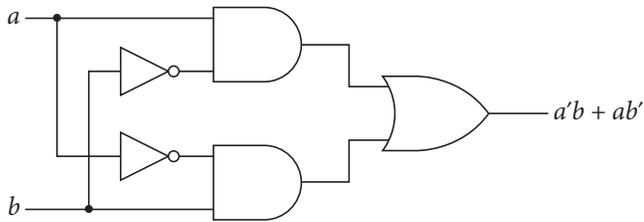
14. (a)



15. (a)



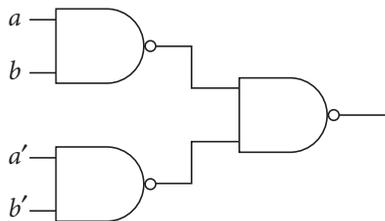
(c)

19. (a) $y(a, b, c) = a'bc + ab'c'$ 20. (a) $y(a, b, c) = (a+b+c)(a+b+c')(a+b'+c)(a'+b+c')(a'+b'+c)$
 $(a' + b' + c')$ 21. (a) $ab + a'b$ (d) $a'b + ab$

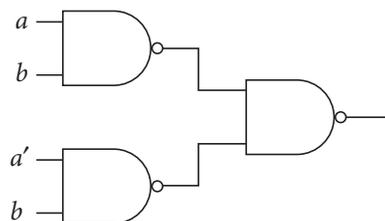
a	b	21(a)	21(d)
0	0	1	0
0	1	0	1
1	0	0	0
1	1	1	1

22.

(a)

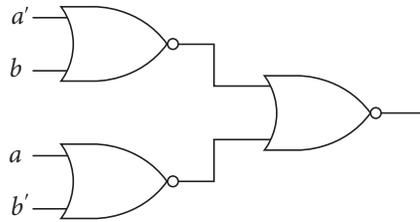


(b)

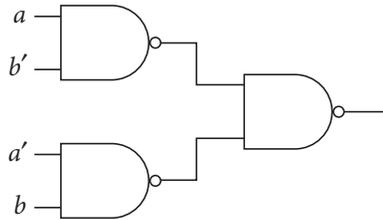
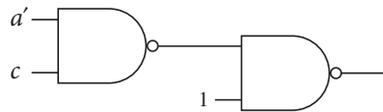
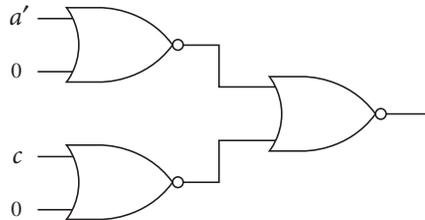
23. (b) $(a' + b)(a + b')$

a	b	
0	0	1
0	1	0
1	0	0
1	1	1

24. (b)



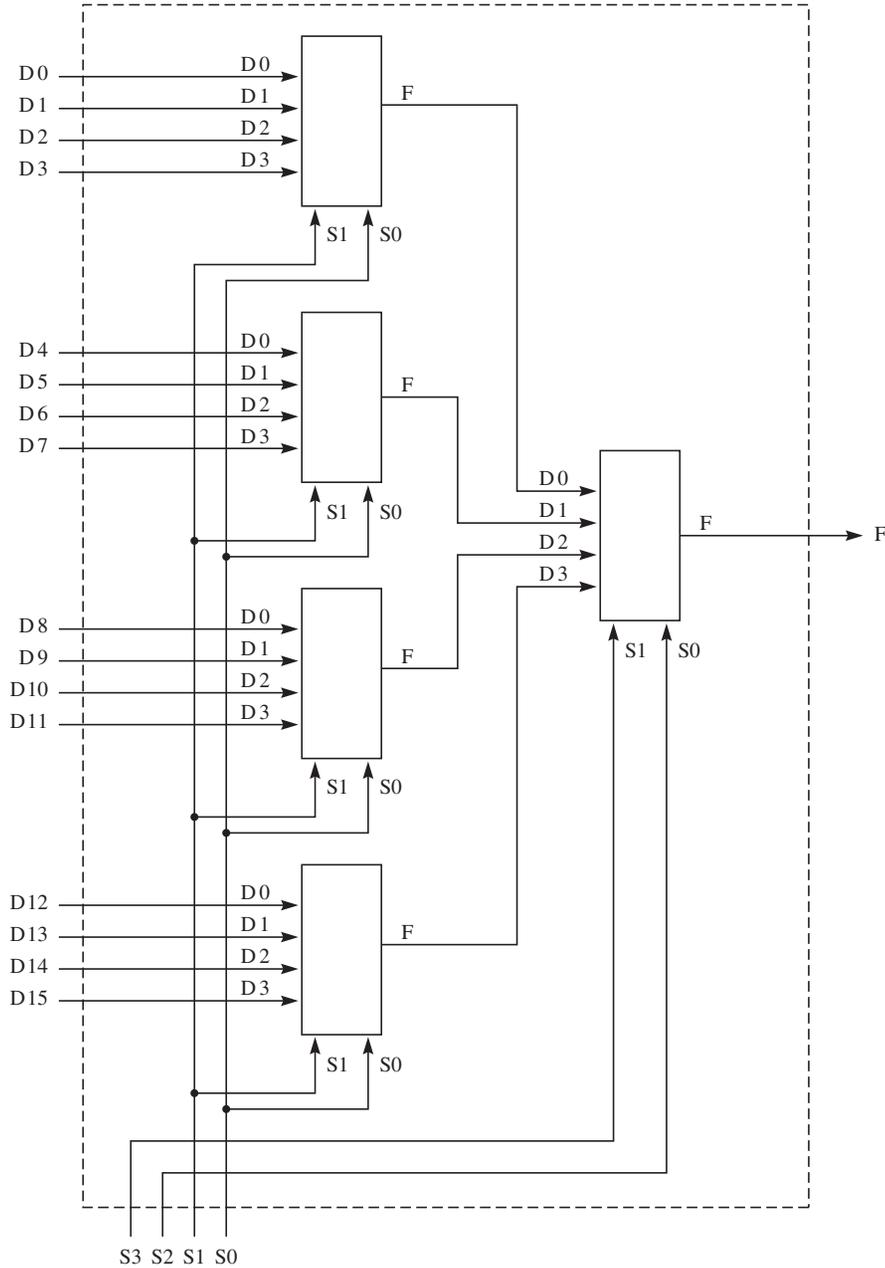
25. (a)

27. (a) $\Sigma(0,3)$ (d) $\Sigma(1,3)$ 28. (b) $\Pi(1,2)$ 29. (a) $x = a'c$ 30. (a) $x = (a')(c)$ 31. (a) $x(a, b, c) = ac + b'c'$ 32. (a) $x(a, b, c) = \Pi(1, 2, 3, 6) = (a + c')(b' + c)$ 33. (a) $x(a, b, c, d) = bc' + a'b'c + b'cd'$ 34. (a) $\Pi(0, 1, 6, 7, 8, 9, 11, 14, 15)$, $x(a, b, c, d) = (b + c)(b' + c')$
 $(a' + c' + d')$ or $x(a, b, c, d) = (b + c)(b' + c')(a' + b + d')$ 35. (a) $x(a, b, c) = a'b' + ab$

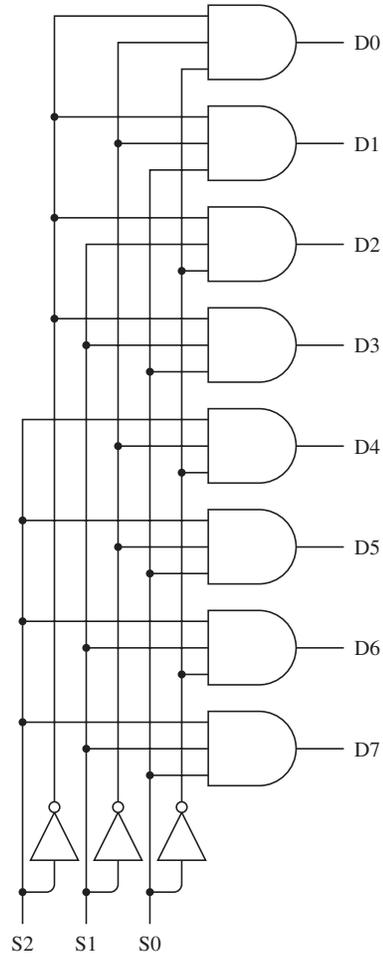
36. (a) $x(a, b, c, d) = bc + bd$

38. (a) The control line acts as an enable that passes the data through unchanged when it is 0. When the control line is 1, it disables the output, which is set to 1, regardless of the data input.

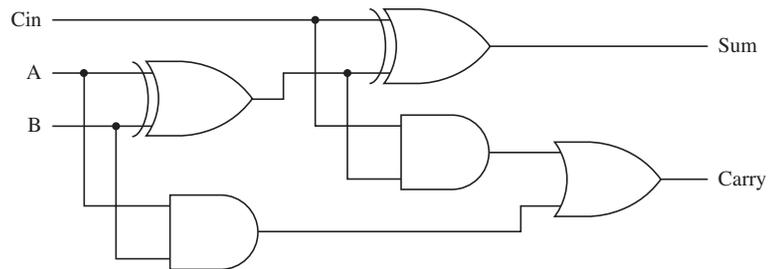
40.



42. (a)



46. (a)



(b) Maximum of three gate delays.

47. (b) Hint: If you look at Figure 10.52 and use the fact that a full adder has three gate delays and a half adder has one gate delay, you might conclude that the total gate delay is 10. However, it is less than that.

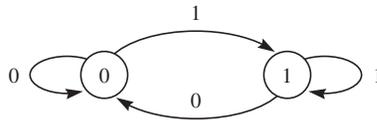
Chapter 11

1. The network will be stable if there is an even number of inverters.

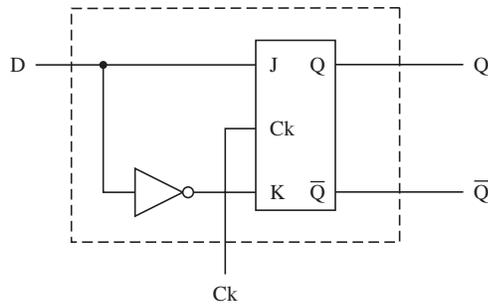
3.

	A	B	C	D	E	R2	S2	Q	\bar{Q}
(a)	0	0	1	1	0	1	0	0	1
(b)	0	0	0	0	0	1	0	0	1

5. (b)



8. (a)



13 (a)

$$DA = A \bar{X}1 + \bar{A} X1 + A B \bar{X}2 + B X1 X2$$

$$DB = \bar{B} \bar{X}1 + B X1 X2 + \bar{A} B X1$$

$$Y = \bar{A} X2 + A X1 X2$$

15 (a)

$$JA = \bar{B} \bar{C} \bar{X} + B C X$$

$$KA = \bar{B} \bar{C} \bar{X} + B C X$$

$$JB = \bar{C} \bar{X} + C X$$

$$KB = \bar{C} \bar{X} + C X$$

$$JC = 1$$

$$KC = 1$$

Chapter 12

5. Hint: (a) Using 16 billion GiB equal to 16×1024^6 bytes, the chip would be 4.3×4.3 meters square, but you must show your calculation.
(b) Yes, but you must explain.

12. (a)

```
lw $t0,20($s6) # Register $t0 gets b[5]
add $t0,$s5,$t0 # Register $t0 gets g + b[5]
sw $t0,16($s4) # a[2] gets g + b[5]
```

(b)

```
100011 10110 01000 0000000000010100
000000 10101 01000 01000 00000 100000
101011 10100 01000 0000000000010000
```

Index

A

- absorption property, 583. *See also* Boolean algebra
- abstract data types (ADTs), 549–550, 626
- abstraction, 4–11
 - in art, 5–6
 - assembly level of, compiling to
 - branching instructions and flow of control, 294–310
 - dynamic memory allocation, 356–377
 - function calls and parameters, 310–336
 - indexed addressing and arrays, 336–356
 - stack addressing and local variables, 288–294
 - in computer systems, 10–11, 774
 - of data, 357
 - definition of, 4
 - in documents, 7–8
 - graphic representations of, 4–5
 - hierarchy diagrams, 5
 - level diagrams, 5
 - nesting diagrams, 5
 - at Level LG1, 626
 - in machines, 9
 - in organizations, 8–9
 - of program control, 357
- activation record. *See* stack frame
- actual parameter, 75
- add immediate instruction, 707–709
- add instruction, 197–198
- ADDA instruction, Pep/9, 723–725
- adder, 613–615
 - full, 613
 - half, 613
 - ripple-carry, 614–615
- adder/subtractor, 615–617
- addition
 - 16-bit with two 8-bit, 622–623
 - signed, 622–624
 - unsigned, 124–125
- address decoding, 674–680
- address lines, 669–670
- address operator, 61
- addressing modes, 745–749
 - MIPS, 745–749
 - base, 745, 752, 760
 - immediate, 745, 747
 - PC-relative, 745, 747–748
 - pseudodirect, 745, 748–749
 - register, 745, 747
 - Pep/9, 702
 - direct, 247–248, 486, 703–704
 - immediate, 247–248, 250, 487, 707–708
 - indexed, 339–340, 353, 752
 - indirect, 362, 486, 709–712
 - letters for, 235
 - problem of computation, 248
 - stack-deferred indexed, 349–350, 376–377
 - stack-indexed, 343
 - stack-relative, 289–292, 486
 - stack-relative deferred, 328, 332, 366
 - trap assertion, 480–482
 - trap operand computation, 483–486
 - .ADDRSS pseudo-op, 238, 353
- ADDSF instruction, 289
- adjacent minterms, 599–600
- Advanced Micro Devices (AMD), 39
- Advanced RISC Machines (ARM), 39
- Aiken, Howard H., 118
- Algol 60, 308
- algorithms, 17–18
- alignment patterns, QR code, 29–30
- allocation techniques, 552–554
- alphabet, 393, 395
 - closure of, 394
 - .ALIGN pseudo-op, 238, 723
- ALU, 617–626
- American Standard Code for Information Interchange. *See* ASCII
- AND Gate, 586–587
- and instructions, 199–200
- AND-OR circuit, 594–597
- App7, 10–11
- application programs, at Level App7, 11
- applications software, 18–19
- architecture, x86, 219–220
- arithmetic infix expression, 401
- arithmetic operators, 139–141
- arithmetic logic unit. *See* ALU
- arithmetic shift left. *See* ASL
- arithmetic shift right. *See* ASR
- arrays, 70–72, 337. *See also* parameters
 - global, 337–340
 - local, 341–344
 - MIPS, 753
- art, abstraction in, 5–6
- ASCII, 23
 - characters, 146–149
 - .ASCII pseudo-op, 238–240
- ASL operations, 139–140
- ASLR instruction, 274–275
- Asmb5, 10
 - assignment statement at, 270
 - Boolean types at, 333–335
 - call-by-reference parameters
 - with global variables at, 324–326
 - with local variables at, 329–330

- Asmb5 (*continued*)
 - call-by-value parameters
 - with global variables at, 313–317
 - with local variables at, 317–320
 - do loop at, 302–303
 - for loop at, 304
 - function call at, 310–313
 - non-void, 320–323
 - global arrays at, 337–338
 - global pointers at, 357–363, 361
 - global structures at, 368–370
 - if statement at, 296–297
 - if/else statement at, 298–299
 - increment statement at, 270
 - linked data structures at, 372–377
 - local arrays at, 341–342
 - local pointers at, 363–367
 - parameter arrays at, 345–347
 - procedure call at, 311
 - switch statement at, 351–352
 - types of statements at, 234
 - while loop at, 300–301
 - ASR
 - instruction, 712–713
 - operations, 139–141
 - ASRR instruction, 274–275, 728
 - assembler, 234–246
 - cross, 246
 - instruction mnemonics, 234–238
 - nonunique nature of inverse mapping of, 255
 - one-to-one mapping of, 255
 - pseudo-operation, 238
 - resident, 246
 - using Pep/9, 244–246
 - assembly language
 - assembler, 234–246
 - cross, 246
 - instruction mnemonics, 234–238
 - pseudo-operation, 238
 - using Pep/9, 244–246
 - comments, 238
 - DECI, DECO, and BR instruction, 248–251
 - disassembler, 255–257
 - HEXO instruction, 252–255
 - immediate addressing, 247–248
 - at Level Asmb5, 10
 - line-oriented nature of, 238
 - program, 305
 - STRO instruction, 251–252
 - symbols, 257
 - program with, 258–260
 - von Neumann illustration, 260–261
 - translating from Level HOL6, 262–279
 - constants and .EQUATE, 275–279
 - global variables and assignment statement, 267–271
 - pep/9 symbol tracer, 273–274
 - placement of instruction and data, 279
 - printf () function, 263–265
 - shift and rotate instruction, 274–275
 - type compatibility, 271–273
 - variables and types, 266
 - assembly line analogy, 761–762
 - assignment operator, 61
 - assignment statements, 59–62, 267–271, 277
 - associative law, 581. *See also* Boolean algebra
 - asynchronous interrupt, 502–503
 - Atanasoff, John V., 118
 - attributes, definition of, 42
- B**
- backslash prefix, 239–240
 - Backus Naur Form. *See* BNF
 - bandwidth
 - of channel, 26
 - equation, 27
 - base addressing, format for, 747
 - base conversions
 - hexadecimal representation, 143–146
 - two's complement binary representation, 130–131
 - unsigned binary representation, 121–123
 - base register, 535. *See also* registers
 - basis, definition of, 82
 - Bélády's anomaly, 548
 - best-fit algorithm, 537–539
 - bidirectional buses, 667–668
 - big-endian order, 204–206
 - binary addition rules, 124
 - binary decoder, 611–612
 - binary digit, 119. *See* bit
 - binary fractions, 154–156
 - binary prefixes, 24–25
 - binary representation
 - two's complement, 125–136
 - base conversions, 130–131
 - negative and zero bits, 135–136
 - number line, 131–133
 - overflow bit, 133–135
 - two's complement range, 128–129
 - unsigned, 118–125
 - base conversions, 121–123
 - binary storage, 118–119
 - carry bit, 125
 - integers, 119–121
 - range for unsigned integers, 123–124
 - unsigned addition, 124–125
 - binary storage, unsigned binary representation, 118–119
 - binary to decimal conversion, two's complement binary representation, 130–131
 - binomial coefficient function, 89–94
 - bit, 119
 - carry, 125
 - definition of, 22
 - hidden, 158–159
 - least significant, 121
 - most significant, 125
 - negative, 135–136
 - overflow, 133–135
 - zero, 135–136
 - bit depth, 33
 - bit-interleaved parity, 567–568
 - black and white images, 32–34
 - accuracy of, 33
 - binary storage, 33–34
 - number of pixels, increasing, 33
 - black box, 579
 - block-interleaved distributed parity, 569
 - block-interleaved parity, 568–569
 - .BLOCK pseudo-op, 238, 241–242, 292
 - BNF, 405
 - Bohm, Corrado, 308–309
 - Boolean algebra, 579–588, 580–581
 - absorption property, 583
 - associative law, 581
 - axioms, 583
 - complements, 585
 - consensus theorem, 583
 - De Morgan's law, 583–584
 - distributive law, 581
 - duality, 581
 - expressions and logic diagrams, 589–591
 - idempotent property, 582–583
 - theorems, 582–583
 - and truth tables, 591–593
 - zero theorem, 583
 - Boolean expressions
 - and logic diagrams, 589–591
 - and truth tables, 591–593
 - Boolean operators, 66
 - Boolean types, 333–336

boundary registers, 535, 762. *See also* registers

BR instruction, 248–251

Brahe, Tycho, 168

BRC instruction, 295

BREQ instruction, 295

BRGE instruction, 295

BRGT instruction, 295

BRLE instruction, 295

BRLT instruction, 295

BRNE instruction, 295

BRV instruction, 295

bugs, von Neumann, 214

. BURN pseudo-op, 238, 471

buses, 667–669

- bidirectional, 667–668
- protocols, 705
- tri-state buffers, 668–669
- unidirectional, 667
- width, 717–722
- wired-or property, 668

byte, 119, 186

- definition of, 23
- compiler, 262–266
- compiler, optimizing, 298
- memory model, heap, 363

instructions

- load, 202–203
- store, 202–203

. BYTE pseudo-op, 238, 242–244

C

C bit, 134–135, 617, 694–695

C compiler, 56–57

- in Microsoft's Visual Studio, 354

C language, 55–108

- alphabet, 393
- context-sensitivity of, 407–408
- dynamic memory allocation, 102–108
- flow of control, 65–72
- functions, 72–81
- identifiers, grammar for, 396–398
- recursion, 81–100
- small subset of, 402–407
- variables, 56–65

C memory model, 57–59

- dynamically allocated variables, 58
- function call, 58
- function return, 58
- global variables, 58
- local variables, 58
- push and pop operations, 58

cache hit, 730

cache memories, 729–738

- direct-mapped, 732–734
- fully associative, exercise 12.10, 777
- locality of reference, 730
- MIPS, 755–757
- set-associative, 734–736
- write policies, 736–737

cache miss, 730

CALL instruction, 312–313

call-by-reference parameters, 76–81. *See also* parameters

- with global variables, 323–328
- with local variables, 329–332

call-by-value parameter, 72–75. *See also* parameters

- with global variables, 314–318
- with local variables, 317–320

calling protocol, 360–361

canonical expressions, 598–599

carry bit, 125. *See also* C bit

central processing unit (CPU), 12–13, 185–186

- control section, 713–716
- data section, 692–696
- MIPS, 755–760
- Pep/9, 692–696

channels

- bandwidth of, 26
- definition of, 26, 40

character constants, 247

character count indicator, QR code, 31

character input program, von Neumann machines, 214–216

character output program, von Neumann machines, 208–214

character representations

- ASCII characters, 146–149
- Unicode characters, 149–153

characteristic tables

- for D flip-flop, 653
- excitation tables versus, 650
- for JK flip-flop, 651
- for SR flip-flop, 647–648

CISC, 39

- versus RISC, 739–741

clock period, 641–642

clock pulse, 641–642

clocked SR flip-flop, 641–643

- block diagram of, 642
- implementation of, 642
- timing diagram of, 643

closure of alphabet, 394

code, 119

- distance, 557
- file, 56

- generation, 406, 435–456
- language translator, 435–455
- parser characteristics, 455–456

generator, 392, 435–455

- point, 149–150
- requirements, 556–559
- word, 556

color depth, 36

color images, 34–38

- color pixels, 35–38
- cone cell sensitivity, 35
- wavelength of light, in visible spectrum, 35

combinational circuits, 578–579

combinational analysis, 588–597

- Boolean expressions and logic diagrams, 589–591
- truth tables and Boolean expressions, 591–593
- two-level circuits, 593–595
- ubiquitous NAND, 595–597

combinational design, 597–609

- canonical expressions, 598–599
- don't-care conditions, 608–609
- dual Karnaugh maps, 607–608
- four-variable Karnaugh maps, 604–607
- three-variable Karnaugh maps, 599–604

combinational devices, 609–626

- abstraction at Level LG1, 626
- adder, 613–615
- adder/ subtractor, 615–617
- arithmetic logic unit, 617–626
- binary decoder, 611–612
- demultiplexer, 612–613
- multiplexer, 610–611
- viewpoints, 609–610

logic gates, 581–588

- alternate representations, 587–588
- Boolean algebra, 580–581
- Boolean algebra theorems, 582–583
- combinational circuits, 579
- logic diagrams, 585–587
- proving complements, 583–585
- truth tables, 580

compaction, 538

compilation process, 420–421

compilers, optimizing, 298

complements. *See* Boolean algebra

complex instruction set computer. *See* CISC

- computer architecture, 183–226
 - direct addressing, 193–206
 - add instruction, 197–198
 - and or instructions, 199–200
 - big-endian versus little-endian orders, 204–206
 - input and output devices, 203–204
 - invert and negate instructions, 200–201
 - load byte and store byte instructions, 202–203
 - load word instruction, 194–195
 - stop instruction, 194
 - store word instruction, 196
 - subtract instruction, 198–199
 - hardware, 184–193
 - central processing unit, 185–186
 - data and control, 189
 - input/output devices, 188
 - instruction format, 189–193
 - main memory, 186–188
 - programming at Level ISA3, 221–226
 - Pep/9 operating system, 222–225
 - read-only memory, 221–222
 - using Pep/9 system, 225–226
 - von Neumann machines, 206–220
 - bugs, 214
 - character input program, 214–216
 - character output program, 208–214
 - decimal to ASCII conversion, 216–217
 - execution cycle, 206–208
 - self-modifying program, 217–220
 - computer chips, 39
 - computer organization
 - big picture, 772–774
 - Level-ISA3 machine
 - add immediate instruction, 707–709
 - arithmetic shift right instruction, 712–713
 - bus protocols, 705
 - CPU control section, 713–716
 - CPU data section, 692–696
 - load word indirect instruction, 709–712
 - store byte direct instruction, 703–704
 - store word direct instruction, 706–707
 - von Neumann cycle, 696–703
 - MIPS machine, 755–760
 - addressing modes, 745–749
 - computer organization, 755–760
 - instruction set, 749–755
 - pipelining, 760–771
 - register set, 743–744
 - performance, 716–717
 - cache memories, 729–738
 - data bus width, 717–722
 - memory alignment, 722–727
 - n*-bit computer, definition of an, 727–729
 - RISC versus CISC, 739–741
 - system performance equation, 738–739
 - simplifications in model, 771–772
 - computer systems, 3–46
 - database systems, 40–46
 - digital information, 22–38
 - hardware, 11–17
 - levels of abstraction, 4–11
 - software, 17–22
 - concatenation, 393–394
 - concurrent processes, 501–516
 - conditional branch instructions, 295
 - conjunction operations, 138
 - consensus theorem, 583. *See also* Boolean algebra
 - constants, 275–279
 - context-free grammars, 400
 - context-sensitive grammar, 399–400, 407–408
 - contiguous allocation, 552
 - control characters, 147–148
 - control lines, 670–672
 - conversion between bases. *See also* conversions
 - core memory. *See* main memory
 - cores, definition of, 40
 - cost of recursion, 99–100
 - CPWA instruction, 299
 - CPWx instruction, 298
 - CR-LF convention, 148
 - critical section, 508–509
 - with semaphore, 513–516
 - CSMux, 694–695
- D**
- D flip-flop, 653–654
 - block diagram, 653
 - characteristic table for, 653
 - constructing, from SR flip-flop, 654
 - implementation of, 654
 - timing diagram for, 653
 - dark module, QR code, 29
 - data and control, 189
 - data bits, QR code, 31–32
 - data bus width, 717–722
 - data file, 19
 - data forwarding, 769. *See also* pipelining
 - database management system (DBMS). *See* database systems
 - database systems
 - language, structure of, 45–46
 - queries, 42–45
 - relations, 41–42
 - De Morgan's law, 583–584
 - for three variables, 584–585
 - deadlock
 - definition of, 513
 - Peterson's algorithm avoiding, 513
 - policy, 519
 - resource allocation graphs, 517–519
 - DECI instruction, 237, 248–251, 488–501
 - decimal prefixes, 24–25
 - decimal to ASCII conversion, 216–217
 - decimal to binary conversion
 - floating-point representation, 154–155
 - two's complement binary representation, 130
 - unsigned binary representation, 123
 - DECO instruction, 237, 248–251
 - decoder, 611. *See also* binary decoder
 - decompression chamber analogy, 647
 - defragmenting, 554
 - delay flip-flop. *See* D flip-flop
 - demand paging, 545–546
 - De Morgan's law. *See* Boolean algebra
 - demultiplexer, 612–613
 - denormalized numbers, representation of, 162–163
 - derivations, in grammar, 397, 406
 - detect and recover policy, of deadlock, 519
 - deterministic FSM. *See* finite state machines
 - digital information, 22–38
 - images, 32–38
 - quantifying space, 22–25
 - quantifying time, 25–27
 - quick response codes, 27–32
 - digraph, 408
 - Dijkstra, Edsger W., 309–310
 - direct addressing, 193–206, 247–248, 486. *See also* addressing modes
 - add instruction, 197–198
 - and or instructions, 199–200
 - big-endian versus little-endian orders, 204–206
 - input and output devices, 203–204
 - invert and negate instructions, 200–201

- load byte and store byte instructions, 202–203
 - load word instruction, 194–195
 - stop instruction, 194
 - store word instruction, 196
 - subtract instruction, 198–199
 - direct-code parser, 423–426
 - direct-mapped cache, 732–734
 - Direct Media Interface (DMI), 40
 - direct memory access (DMA), 17, 189, 772
 - directed graph, 408
 - dirty bit, 546–547
 - disassembler, 255–257
 - disjunction operations, 138
 - disk, 16–17
 - disk drives, 550–551
 - DisplayPort, 40
 - distributive law, 581. *See also* Boolean algebra
 - divide and conquer strategy, 86
 - do loop, 69–70, 302–303
 - nonatomic nature of, 510
 - document
 - abstraction in, 7–8
 - file, 19
 - domains, definition of, 42
 - don't-care conditions, 608–609. *See also* Karnaugh maps
 - dual Karnaugh maps, 607–608
 - duality. *See* Boolean algebra
 - dynamic branch prediction, 766–767. *See also* pipelining
 - dynamic memory, 673
 - dynamic memory allocation, 102–108, 356–377
 - global pointers, 357–363
 - linked data structures, 105–108
 - local pointers, 363–367
 - operators that control, 102
 - pointers, 102–104
 - structures, 104–105
 - global, 367–372
 - linked data, 372–377
- E**
- Eckert, J. Presper, 118, 206
 - edge-triggered flip-flops, 644
 - electrically erasable PROM (EEPROM), 674
 - Electronic Delay Storage Automatic Calculator (EDSAC), 206
 - Electronic Numerical Integrator and Calculator (ENIAC), 118, 206
 - elements of array, reversing, 94–95
 - ellipses, 169
 - empty string, 394
 - empty transitions, 412–415
 - algorithm to remove, 413–415
 - as nondeterministic, 413
 - enable lines, definition of, 609
 - . END pseudo-op, 238–240
 - . EQUATE pseudo-op, 238, 275–279, 292–294
 - erasable PROM (EPROM), 674
 - error-correcting codes, 557–562
 - error-detecting codes, 555–557
 - excess representations, 156–157
 - excitation tables, 655
 - characteristic tables versus, 650
 - construction of, 650
 - for D flip-flops, 655
 - for JK flip-flops, 655
 - for SR flip-flop, 649
 - for T flip-flops, 655
 - exclusive OR. *See* XOR
 - execution cycle, von Neumann Machines, 206–208
- F**
- factorial function, 82–85
 - feedback circuits, 638
 - stable state, 639
 - unstable state, 638
 - field, definition of, 104
 - FIFO page-replacement algorithm, 547–549
 - file abstraction, 551
 - file management, 549–550
 - allocation techniques, 552–554
 - disk drives, 550–551
 - file abstraction, 551
 - files
 - management of, 19–20
 - types of information, 19
 - finder patterns, QR code, 28–29
 - finite-state machines (FSMs), 392, 408–435, 771, 773–774
 - concept of, 773–774
 - in DECI, 488
 - deterministic, 414, 448–449
 - with empty transitions, 412–415
 - grammars versus, 417–418
 - implementation of, 418–435
 - compilation process, 420–421
 - direct-code parser, 423–426
 - input buffer class, 426–428
 - multiple-token parser, 428–435
 - table-lookup parser, 421–423
 - multiple token recognizers, 415–417
 - nondeterministic, 410–412
 - to parse identifier, 409–410
 - simplified, 410
 - first-fit algorithm, 538
 - first fit versus best fit, 539
 - first-in, first-out (FIFO), 547–549
 - fixed-partition multiprogramming, 533–534
 - flash memory, 674
 - flip-flop, 642
 - floating-point division, 63–64
 - floating-point representation, 153–168
 - binary fractions, 154–156
 - excess representations, 156–157
 - hidden bit, 158–159
 - IEEE 754 floating-point standard, 165–168
 - special values, 159–165
 - flow of control, 65–72
 - arrays, 70–72
 - different styles of, 307
 - do loop, 69–70
 - in early languages, 307–308
 - for loop, 70–72
 - if/else statement, 66–67
 - structured, 307
 - switch statement, 67–68
 - while loop, 68–69
 - for loop, 70–72, 303–305
 - formal parameter, 75
 - format information area, QR code, 29
 - format trace tags, 273, 294, 339. *See also* trace tags
 - Fortran, 307–309
 - four-variable Karnaugh maps, 604–607
 - fractions, binary, 154–156
 - fragmentation, 537–538
 - full adder, 613. *See also* adder
 - function calls, 310–313
 - non-void, 320–323
 - function SE(im), 747
 - functions, 72–81. *See also* parameters
 - call-by-reference parameters, 76–81
 - call-by-value parameters, 72–75
 - call mechanism, 310–332
 - void, 313
 - void functions, 72–75
- G**
- gate delay, 593–594
 - global arrays, 337–340
 - memory allocation for, 339
 - translation rules for, 340

global pointers, 357–363
 global structures, 367–372
 global variables, 59–62, 267–271. *See also* variables
 address operator, 61
 assignment operator, 61
 call-by-reference parameters with, 323–328
 call-by-value parameters with, 313–317
 local versus, 292
 memory model for, 62
 glyph, 149
 goto controversy, 309–310
 goto statement, 307–308
 gradual underflow, 162
 grammars, 392, 395–396
 for C identifiers, 396–398
 for C language, 402–407
 context-sensitive, 399–400, 407–408
 for expressions, 401–402
 finite-state machines versus, 417–418
 four parts of, 395
 for signed integers, 398–399
 grayscale images, 32–34
 accuracy of, 33
 binary storage, 33–34
 number of pixels, increasing, 33

H

half adder, 613. *See also* adder
 hamming distance, 556–557
 hardware, 11–17
 central processing unit, 12–13
 concurrency, 702
 disk, 16–17
 main memory, 13–16
 head crash, 551
 heap pointer, 359–360
 hexadecimal representation, 142–153
 base conversions, 143–146
 conversion chart, 144
 HEXO instruction, 237, 252–255, 498–501
 hidden bit, 158–159
 hierarchy diagrams, abstraction, 5
 leaf, 5
 node, 5
 root, 5
 High-Definition Multimedia Interface (HDMI), 40
 high-order languages, at Level HOL6, 10

HOL6, 10, 262–279
 Boolean types at, 333
 call-by-reference parameters
 with global variables at, 324
 with local variables at, 329
 call-by-value parameters
 with global variables at, 314
 with local variables at, 318
 do loop at, 302–303
 for loop at, 304
 function call at, 311
 non-void, 320–322
 global arrays at, 337
 global pointers at, 357, 363
 global structures at, 368, 370
 if statement at, 300
 if/else statement, 299
 illegal and legal at, 272
 information representation at, 118, 169–170
 linked data structures at, 372, 375
 local arrays at, 341
 local pointers at, 363, 365
 logical operators, truth tables for, 138
 parameter arrays at, 345
 procedure call at, 311
 switch statement at, 351
 while loop at, 300–301

I

IBM 360 family, 727
 idempotent property, 582–583. *See also* Boolean algebra
 identifiers, definition of, 45
 identity element, 394
 IEEE 754 floating-point standard, 165–168
 if statement, 295–297
 if/else statement, 66–67, 298–300
 ignore policy, of deadlock, 519
 images, 32–38
 accuracy of, 33
 binary storage, 33–34
 black and white, 32–34
 color, 34–38
 grayscale, 32–34
 number of pixels, 33
 immediate addressing, 247–248, 250, 487.
 See also addressing modes
 format for, 747
 index, 554
 indexed addressing and arrays, 336–356.
 See also addressing modes
 arrays passed as parameters, 344–350
 global arrays, 337–340
 local arrays, 341–344
 switch statement, 350–354
 indirect addressing, 362, 487. *See also* addressing modes
 infinity, representation of, 161
 information representation, 117
 character representations
 ASCII characters, 146–149
 Unicode characters, 149–153
 floating-point representation, 153–168
 binary fractions, 154–156
 excess representations, 156–157
 hidden bit, 158–159
 IEEE 754 floating-point standard, 165–168
 special values, 159–165
 hexadecimal representation, 142–153
 base conversions, 143–146
 models, 168–170
 operations in binary, 136–142
 arithmetic operators, 139–141
 logical operators, 137–138
 register transfer language, 138–139
 rotate operators, 141–142
 two's complement binary
 representation, 125–136
 base conversions, 130–131
 negative and zero bits, 135–136
 number line, 131–133
 overflow bit, 133–135
 two's complement range, 128–129
 unsigned binary representation,
 118–125
 base conversions, 121–123
 binary storage, 118–119
 carry bit, 125
 integers, 119–121
 range for unsigned integers, 123–124
 unsigned addition, 124–125
 input buffer class, 426–428
 input/output (I/O) devices, 188
 direct addressing, 203–204
 instruction format, 189–193
 instruction register, 14
 instruction set
 MIPS, 749–755
 Pep/9 at Asmb5, 236–237
 instruction specifier, 189, 191
 integer division, 63–64
 integers
 conversions. *See* base conversions
 signed, 398–399
 unsigned binary representation, 119–121

integrated development environment (IDE), 100–101
 integrated memory controller (IMC), 40
 Intel Core i7 system, 39–40
 internal fragmentation, 542
 interpretation, advantages and disadvantages of, 420
 invert instructions, 200–201
 inverter, 585–587
 I/O completion, 502–503
 ISA3, 10
 construction of
 add immediate instruction, 707–709
 arithmetic shift right instruction, 712–713
 bus protocols, 705
 CPU control section, 713–716
 CPU data section, 692–696
 load word indirect instruction, 709–712
 store byte direct instruction, 703–704
 store word direct instruction, 706–707
 von Neumann cycle, 696–703
 information representation at, 118, 169–170
 logical operators, truth tables for, 137
 Pep/9 instruction set at, 190
 programming at, 221–226
 Pep/9 operating system, 222–225
 read-only memory, 221–222
 using Pep/9 system, 225–226
 types of bit patterns at, 234

J

Jacopini, Giuseppe, 308
 Java Virtual Machine (JVM), 419
 JK flip-flop, 650–653
 block diagram of, 651
 characteristic table for, 651
 constructing, from SR flip-flop, 651–652
 design tables, constructing, 651–652
 implementation of, 653
 jump register instruction, 749
 jump table, 353

K

Karnaugh maps, 599–609
 don't care conditions, 608–609

dual, 607–608
 equivalent of idempotent property, 602–603
 four-variable, 604–607
 three-variable, 599–604
 Kepler, Johannes, 168–169

L

language, 394–395
 structure of, 45–46
 translator, 435–455
 language translation principles
 code generation, 435–456
 language translator, 435–455
 parser characteristics, 455–456
 concatenation, 393–394
 finite-state machines, 408–435
 with empty transitions, 412–415
 grammars versus, 417–418
 implementation of, 418–435
 multiple token recognizers, 415–417
 nondeterministic, 410–412
 to parse an identifier, 409–410
 simplified, 410
 grammars, 395–396
 for C identifiers, 396–398
 for C language, 402–407
 context-sensitive, 399–400, 407–408
 for expressions, 401–402
 four parts of, 395
 for signed integers, 398–399
 languages, 394–395
 parsing problem, 400–401
 large program behavior, 543
 latches, 638–641
 latency, 16, 551
 LDW_r instruction, 709–712
 least recently used (LRU), 547–549
 least significant bit, 121
 left-symmetric parity
 distribution, 569
 level diagrams, abstraction, 5
 level sensitive flip-flop, 643
 lexical analysis, 420
 lexical analyzer, 447–448, 455
 LG1, 10–11
 linked allocation technique, 553–554
 linked data structures, 105–108, 372–377
 little-endian order, 204–206
 load byte instructions, 202–203
 load word indirect instruction, 709–712
 load word instruction, 194–195
 loader, 470
 invoking, 472
 Pep/9 loader, 472–474
 Pep/9 operating system, 471–472
 program termination, 474–475
 relocatable, 534
 load/store machines, 743
 local arrays, 341–344
 memory allocation for, 343
 translation rules for, 344
 local pointers, 363–367
 local structures, 371–372
 local variables, 62–65, 292–294.
 See also variables
 call-by-reference parameters with, 329–332
 call-by-value parameters with, 317–320
 floating-point division, 63–64
 global versus, 292
 integer division, 63–64
 memory model for, 65
 program with, 293
 locality of reference, 730
 logic diagrams, 585–587, 589
 and Boolean expressions, 589–591
 logic gates
 alternate representations, 587–588
 Boolean algebra, 580–581
 Boolean algebra theorems, 582–583
 combinational circuits, 579
 at Level LG1, 11
 logic diagrams, 585–587
 proving complements, 583–585
 truth tables, 580
 logic programming, 587
 logical addresses, 534–537, 544
 logical operators, 137–138
 LRU page-replacement
 algorithm, 549
 LWI instruction, MIPS, 754
 LW instruction, MIPS, 751

M

machine language, 193
 at Level ISA3, 10
 machine vectors, 223
 advantage of, 225

- machines
 - abstraction in, 9
 - independence, 57
 - main(), 264, 288
 - main memory, 13–16, 186–188, 194
 - malloc() function, 359–361
 - Mark I computer, 118
 - master-slave SR flip-flop, 643–648
 - block diagram of, 645
 - characteristic tables, 647–648
 - implementation of, 645
 - operation of, 645
 - state transition diagram for, 648
 - timing diagram of, 647
 - Matisse, Henri, 5–6
 - Mauchly, John W., 118
 - Mc2, 10–11
 - MDRMux, 694
 - megahertz myth, 770
 - memory
 - access, direct, 189
 - alignment, 722–727
 - hierarchies, 738
 - location and address, distinction
 - between, 187, 188
 - main, 186–188, 289, 291
 - of Pep/9, 471
 - mapping, Pep/9, 221
 - programmable read-only, 222
 - random-access, 222
 - read-only, 221–222
 - read/write, 222
 - volatile, 222
 - memory address, 257
 - memory address register (MAR), 694
 - memory alignment, 722–727
 - memory allocation, 532
 - fixed-partition multiprogramming, 533–534
 - logical addresses, 534–537
 - paging, 540–543
 - uniprogramming, 532–533
 - variable-partition
 - multiprogramming, 537–540
 - memory chip, 669–672
 - control lines, 670, 672
 - read enable box, 673
 - reading from, 670
 - writing to, 670
 - memory data register (MDR), 694
 - memory hierarchy, 738
 - memory management, 19
 - algorithms, 359
 - memory-mapped input/output (I/O), 14, 188
 - memory read bus protocol, 705
 - memory-style ECC, 566–567
 - memory subsystems, 669–674
 - address lines, 669–670
 - control lines, 670–672
 - dynamic memory (DRAM), 673
 - electrically erasable PROM, 674
 - erasable PROM, 674
 - flash memory, 674
 - memory chips, 669–672
 - memory write operation, 672–673
 - monostable multivibrator, 672
 - programmable ROM, 673–674
 - read enable box, 673
 - read-only memory, 673
 - static memory, 673
 - memory write bus protocol, 705
 - metric prefixes, 23
 - microprogramming, at Level Mc2, 11
 - Microsoft Macro Assembler (MASM), 354
 - minterm, 598–599
 - MIPS machine, 743–771
 - addressing modes, 744–749
 - computer organization, 755–760
 - instruction set, 749–755
 - pipelining, 760–771
 - register set, 743–744
 - mnemonics, 234–238
 - mode indicator, QR code, 30–31
 - module, QR code, 28
 - monostable multivibrator, 672
 - Moore, Gordon, 728
 - Moore's law, 728–729
 - most significant bit, 125
 - MOVSPA instruction, 332
 - multi-user operating systems, 470
 - multiple-token parser, 428–435
 - multiple token recognizers, 415–417, 428–435
 - multiplexer, 610–611
 - multiplication algorithm
 - iterative, 383–384
 - recursive, 382
 - multiprocessing, 505–506, 508
 - multiprogramming, 503, 508
 - fixed-partition, 533–534
 - variable-partition, 537–540
 - mutual exclusion
 - first attempt at, 509–510
 - Peterson's algorithm for, 512–513
 - second attempt at, 510–512
 - mutual recursion, 98–99
 - function prototype, 98–99
 - in recursive descent compiler, 99
- ## N
- N bit, 135–136, 194–195, 198–199, 619
 - n*-bit computer, 727–729
 - NaN (not a number), representation
 - of, 161
 - NAND gate, 586–587
 - NAND-NAND circuit, 595–596
 - negate instructions, 200–201
 - negation operation, 126
 - negative bit. *See* N bit
 - negative numbers, convenient property
 - of, 126
 - nesting diagrams, abstraction, 5
 - NetBeans IDE, 101
 - node, definition of, 105
 - nondeterministic FSM. *See* finite state machines
 - non-void function calls, 320–323
 - non-volatile memory, 16
 - nonunary trap, 480
 - NOP instruction, 237, 487
 - NOPl instruction, 237, 479–480
 - NOR gate, 586–587
 - NOR-NOR circuit, 597
 - NOT operation. *See* ones' complement
 - number line, for two's complement
 - binary representation, 131–133
 - nybble, 474
- ## O
- object program, 56
 - one-shot devices. *See* monostable multivibrator
 - ones' complement, 125–126
 - opcode, 189, 193
 - mnemonic for, 234
 - unimplemented, 237
 - operand specifier, 189, 192, 194, 289
 - operating system, 19–20
 - functions of, 19
 - at Level OS4, 10
 - Pep/9, 222–225, 471–472
 - processes in, 503–505
 - purposes of, 470
 - types of, 470
 - vectors, 498–501
 - operation code. *See* opcode
 - OprndSpec, 343–344
 - optimizing compiler, 297–298
 - advantages and disadvantages
 - of, 298
 - purpose of, 298
 - OR-AND circuit, 594–596
 - OR gate, 585–587

or instructions, 199–200
 organizations, abstraction in, 8–9
 ori instruction, MIPS, 754
 OS4, 10
 overflow bit, 133–135. *See also* V bit

P

Pac-Man effect, 601
 page fault, 545
 page replacement, 546–547
 algorithms, 547–549
 page table, 541
 paging, 540–543
 parallelism, 717
 parameters
 arrays, 344–350
 run-time stack for, 348
 translation rules for, 350
 call-by-reference
 with global variables, 323–328
 with local variables, 329–332
 call-by-value
 with global variables, 313–317
 with local variables, 317–320
 parity bit, 555
 parser, 392, 455–456
 characteristics, 455–456
 direct-code, 423–426
 multiple-token, 428–435
 recursive descent, 456
 table-lookup, 421–423
 parsing, 420
 problem, 400–401
 Pascal's triangle, 89
 PC-relative addressing, format for, 748
 PCB, 477, 503–505
 Pep/9 computer, 184
 alphabet, 393
 block diagram, 185
 central processing unit, 185
 input/output devices, 188
 instruction format, 189–193
 instruction set at Level ISA3, 190
 main memory, 186–188
 operating system, 222–225
 programming at Level ISA3, 225–226
 perfect code, 560
 Peripheral Component Interconnect
 Express (PCIe), 40
 Peterson's algorithm, for mutual
 exclusion, 512–513
 pipelining, 760–771
 data forwarding, 769
 dynamic branch prediction, 766–767

hazards, 762, 764
 controlling, 762, 765
 RAW data, 767–768
 WAR data, 770
 instruction reordering, 768
 MIPS machine, 760–771
 superscalar machines, 769–770
 pixels, 15
 Platform Controller Hub (PCH), 40
 pointers, 102–104
 as addresses, 361
 assignment, 102
 global, 357–363
 local, 363–367
 stack. *See* SP
 prevention policy, of deadlock, 519
 printBar(), 317
 printf() function, translating, 263–265
 procedures, definition of, 72
 process
 concurrent, 501–516
 cooperating, 506
 definition of, 477
 process control block. *See* PCB
 process management
 concurrent process, 501
 asynchronous interrupt, 502–503
 critical section, 508–509
 multiprocessing, 505–506
 mutual exclusion, 509–513
 program, 506–508
 semaphore, 513–516
 loader, 470
 Pep/9 loader, 472–474
 Pep/9 operating system, 471–472
 program termination, 474–475
 trap, 475
 addressing mode assertion,
 480–482
 handler. *See* trap, handler
 mechanism, 476–477
 operand address computation,
 483–486
 RETR instruction, 477
 processor management, 19
 production
 in grammar, 397
 recursive, 399
 program, 506–508
 definition of, 18
 file, 19
 statements alignment, 724–727
 termination, 474–475
 programmable read-only memory
 (PROM), 222, 673–674

programming at Level ISA3, 221–226
 Pep/9 operating system, 222–225
 read-only memory, 221–222
 using Pep/9 system, 225–226
 pseudodirect addressing, format for,
 748–749

Q

QR code. *See* quick response codes
 queries, 42–45
 relationship with database and
 result, 43
 result of, 42–43
 statements for, 43–45
 queue of PCBs, 503–505
 quick response codes, 27–32
 alignment and format regions,
 28–30
 alignment patterns, 29–30
 dark module, 29
 finder patterns, 28–29
 format information area, 29
 separators, 29
 timing patterns, 29
 version information area, 30
 correction levels in, 31–32
 information bits, 30–32
 character count indicator, 31
 data bits, 31–32
 mode indicator, 30–31
 redundant bits of error
 correction, 31

R

radix, of number system, 122
 RAID storage systems, 562–569
 level 0: nonredundant striped,
 562–563
 level 1: mirrored, 563–564
 level 2: memory-style ECC, 566–567
 level 3: bit-interleaved parity,
 567–568
 level 4: block-interleaved parity,
 568–569
 level 5: block-interleaved distributed
 parity, 569
 levels 01 and 10: striped and
 mirrored, 564–566
 random-access memory (RAM), 16,
 222, 471
 range
 for two's complement, 128–129
 for unsigned integers, 123–124

- RAW data hazard. *See* pipelining
 - read-after-write (RAW) hazard, 767–768
 - read enable box, 672–673
 - read-only memory (ROM), 221–222, 471, 673
 - read/write memory (RWM), 222
 - real-time operating systems, 470
 - recursion, 81–100
 - binomial coefficient function, 89–94
 - cost of recursion, 99–100
 - elements of array, reversing, 94–96
 - factorial function, 82–85
 - mutual recursion, 98–99
 - recursive addition, 87–89
 - recursive definition, 81
 - recursive thinking, 85–86
 - Towers of Hanoi, 95–98
 - recursive addition, 87–89
 - recursive definition, 81
 - recursive descent, 99
 - parser, 456
 - recursive thinking, 85–86
 - divide and conquer strategy, 86
 - macroscopic viewpoints, 85–86
 - microscopic viewpoints, 85–86
 - reduced instruction set computer. *See* RISC
 - redundant bits of error correction, QR code, 31
 - register addressing, format for, 747
 - register sets, 743–744
 - register transfer language. *See* RTL
 - registers, 12, 666–667
 - base and bound, 535
 - block diagram of, 666
 - implementation with D flip-flops, 666–667
 - MIPS, 743–744
 - regular expressions, 392
 - relational database systems, 41–42
 - relational operators, 65
 - relations, definition of, 42
 - relays, 118
 - relocatable loader, 534
 - resource allocation graph, 517–519
 - RET instruction, 315
 - RETTR instruction, 477
 - return, translating, 264
 - RGB stripe, 35
 - ripple-carry adder, 614
 - RISC, 39, 578
 - versus CISC, 739–741
 - ROLr instruction, 274–275
 - RORr instruction, 274–275
 - rotate left operation (ROL), 141
 - rotate operators, 141–142
 - rotate right operation (ROR), 141
 - round to nearest, ties to even rule, 158–159
 - RTL, 138–139, 289, 476
 - operations and symbols, 138
 - specification
 - add instruction, 197
 - arithmetic shift left operation, 140
 - exclusive OR operation, 139
 - and instruction, 199
 - or instruction, 199
 - invert instruction, 200
 - load byte instruction, 202
 - load word instruction, 194
 - negate instruction, 201
 - rotate left operation, 141
 - rotate right operation, 141
 - store byte instruction, 202
 - store word instruction, 196
 - subtract instruction, 198
 - run command, 56
 - run-time stack, 290–292
 - alignment on, 724
 - for call-by-reference parameter
 - with global variable, 328
 - with local variable, 332
 - for parameter arrays, 348
- S**
- seek time, 16, 551
 - selective inverter gate, 610
 - semantics, 392
 - semaphore, 513–516
 - critical sections with, 516
 - separators, QR code, 29
 - sequential access memory, 16
 - sequential circuits, 637–682
 - computer subsystems, 666–682
 - address decoding, 674–680
 - buses, 667–669
 - memory subsystems, 669–674
 - registers, 666–667
 - two-port register bank, 680–682
 - latches and clocked flip-flops, 641–656
 - clocked SR flip-flop, 641–643
 - D flip-flop, 653–654
 - excitation tables, 655
 - JK flip-flop, 650–653
 - master–slave SR flip-flop, 643–648
 - SR latch, 639–641
 - T flip-flop, 654–655
 - preset and clear, 661
 - sequential analysis, 656–661
 - sequential design, 656–665
 - Serial Advanced Technology Attachment (SATA), 40
 - set-associative cache, 734–736
 - set, bit-mapped representation of, 481–482
 - set interpretation of Boolean algebra, 588
 - set theory interpretation, 587–588
 - signed integer
 - grammar for, 398–399
 - structure of, 126
 - significand, 156
 - single-error-correcting codes, 559–562
 - single-user operating systems, 470
 - sll instruction, MIPS, 753
 - software, 17–22
 - analysis and design, 20–22
 - applications software, 18–19
 - distribution
 - advantage of object code for, 256
 - advantage of source code for, 256
 - interrupt, 501
 - operating systems, 19–20
 - systems software, 18–19
 - solid-state disks (SSDs), 17
 - source program, 56
 - space/time tradeoff, 716
 - combinational circuits, 595, 609
 - exercise 10.46, 633
 - spaghetti code, 305–307
 - spatial locality, 730
 - special values, in floating-point representation, 159–165
 - specialized hardware units, MIPS, 755, 757
 - spin lock, 514
 - SR latch, 639–641
 - stable state, in feedback circuits, 638
 - stack-deferred indexed addressing, 349. *See also* addressing modes
 - stack frame, 75
 - stack-indexed addressing, 343. *See also* addressing modes
 - stack pointer (SP)
 - before and after program executes, 291–292
 - as memory address, 289
 - stack-relative addressing, 289–292, 486. *See also* addressing modes
 - deferred, 327–328, 332, 366–367
 - run-time stack, 290–292

- start symbol, 396
 - state registers, 656
 - state transition diagram, 408–409
 - state transition table, 409–410, 412–413
 - state variable, 421
 - static memory (SRAM), 673
 - stop instruction, 194
 - storage management, 532
 - error-detecting and error-correcting codes
 - code requirements, 556–559
 - error-detecting codes, 556–557
 - single-error-correcting codes, 559–562
 - file management, 549–550
 - allocation techniques, 552–554
 - disk drives, 550–551
 - file abstraction, 551
 - memory allocation, 532
 - fixed-partition
 - multiprogramming, 533–534
 - logical addresses, 534–537
 - paging, 540–543
 - uniprogramming, 532–533
 - variable-partition
 - multiprogramming, 537–540
 - RAID storage systems, 562–569
 - level 0: nonredundant striped, 562–563
 - level 1: mirrored, 563–564
 - level 2: memory-style ECC, 566–567
 - level 3: bit-interleaved parity, 567–568
 - level 4: block-interleaved parity, 568–569
 - level 5: block-interleaved distributed parity, 569
 - levels 01 and 10: striped and mirrored, 564–566
 - virtual memory
 - demand paging, 545–546
 - large program behavior, 543
 - page replacement, 546–547
 - page-replacement algorithms, 547–549
 - virtual memory, 544–545
 - store byte instructions, 202–203
 - direct, 703–704
 - store word instruction, 196
 - direct, 706–707
 - STRO instruction, 237, 251–252, 482, 498–501
 - structured flow of control, 307
 - structured programming theorem, 8, 308–309
 - Structured Query Language (SQL), 43
 - structures, 104–105
 - global, 367–372
 - linked data, 372–377
 - local, 371–372
 - SUBSP instruction, 289, 292
 - subtract instruction, 198–199
 - subtractor. *See* adder/subtractor
 - subtraction, binary, 621–624
 - superscalar machines, 769–770. *See also*
 - pipelining
 - supervisor call, 501
 - sw instruction, MIPS, 751, 759–760
 - switch statement, 67–68, 350–354
 - symbol, 257–261, 266
 - program with, 258–260
 - trace tags, 273, 294
 - tracer, 273–274
 - von Neumann illustration, 260–261
 - symbol trace tags. *See* trace tags
 - synchronous interrupt, 501
 - syntax, 392
 - techniques to specify, 392
 - tree, 402
 - for parse of nonterminal alphabet, 407
 - system bus, 12
 - system performance equation, 25–26, 738–739
 - systems software, 18–19
- T**
- T flip-flop, 654–655
 - block diagram of, 655
 - characteristic table for, 655
 - table-lookup parser, 421–423
 - Teletype Model 33, 148
 - temporal locality, 730
 - three-variable Karnaugh maps, 599–604
 - threshold, of gate, 645–646
 - time out, 502
 - time sharing, 502
 - timing diagram
 - of clocked SR flip-flop, 643
 - of D flip-flop, 653
 - of master-slave SR flip-flop, 647
 - for SR latch, 641
 - of von Neumann cycle, 699
 - timing patterns, QR code, 29
 - token, definition of, 415
 - Towers of Hanoi, 95–98
 - trace tags, 273
 - format, 273, 294, 339
 - symbol, 273, 294
 - transmission time, 551
 - trap, 475
 - handler
 - DECI, 488–495
 - DECO, 495–498
 - HEXO and STRO, 498–501
 - no-operation, 487
 - mechanism, 476–477
 - nonunary, 480
 - Pep/9, 476
 - tree diagram. *See* hierarchy diagrams
 - tri-state buffers, 668–669
 - truth table, 580, 588
 - and Boolean expressions, 591–593
 - tuples, definition of, 42
 - two-level circuits, 593–595
 - two-port register bank, 680–682
 - two's complement binary representation, 125–136
 - base conversions, 130–131
 - negative and zero bits, 135–136
 - number line, 131–133
 - overflow bit, 133–135
 - range for, 128–129
 - type compatibility, 271–272
- U**
- ubiquitous NAND, 595–597
 - unary instruction, 189
 - unary operation, 138
 - Unicode character, 149–153
 - Unicode Transformation Format (UTF), 150–153
 - unidirectional buses, 667
 - Unified Modeling Language (UML), 439–442
 - unimplemented opcode. *See* opcode
 - uniprogramming, 532–533
 - Universal Serial Bus (USB), 40
 - unsigned addition, 124–125
 - unsigned binary representation, 118–125
 - base conversion, 121–123
 - binary storage, 118–119
 - carry bit, 125
 - integer, 119–121
 - range for unsigned integer, 123–124
 - unsigned addition, 124–125
 - unstable state, in feedback circuits, 638
 - user stack, 223

V

V bit, 134–135, 614–615, 712–713
 variable-partition multiprogramming,
 537–540
 variables, 56–65
 attributes of, 59
 C compiler, 56–57
 C memory model, 57–59
 global, 267–271
 and assignment statements,
 59–62
 call-by-reference parameters with,
 323–328
 call-by-value parameters with,
 313–317
 local, 62–65, 292–294
 call-by-reference parameters with,
 329–332
 call-by-value parameters with,
 317–320
 machine independence, 57
 types, 266
 version information area, QR code, 30
 virtual computer, 184

virtual memory, 544–545
 demand paging, 545–546
 large program behavior, 543
 page replacement, 546–547
 algorithms, 547–549
 void function, 72–75
 actual parameter, 75
 allocation process for, 74
 deallocation process for, 75
 formal parameter, 75
 stack frame, 75
 volatile memory, 16, 222
 von Neumann execution cycle, 206–208,
 641, 696–703
 MIPS, 744
 Pep/9 at Level ISA3, 250, 312–313
 symbol and, 260–261
 von Neumann machines, 206–220
 bugs, 214
 character input program, 214–216
 character output program, 208–214
 decimal to ASCII conversion, 216–217
 execution cycle, 206–208
 self-modifying program, 217–220

W

WAR data hazard. *See* pipelining
 Watson, Thomas J., 118
 while loop, 68–69, 300–302
 wired-OR property, 668
 .WORD pseudo-op, 238, 242–244
 working set, 543
 write-after-read (WAR) hazard, 770

X

x86
 architecture, 219–220
 assembly language, 261–262
 compiling to, 354–356
 microcode in systems, 742
 XOR gate, 587
 XOR operator, 335–336

Z

Z bit, 135–136, 618, 694–695
 zero bit. *See* Z bit
 zero theorem, 583. *See also* Boolean algebra