

Intel® Pentium® 4
Processor
Extreme Edition

8.5 GB/s

82925XE
MCH

PCI Express®
x16 Graphics

8.0
GB/s

DDR2

8.5 GB/s

DDR2

2 GB/s DMI

Intel® High
Definition Audio

4 PCI
Express® x1

8 Hi-Speed
USB 2.0 Ports

500
MB/s

60
MB/s

ICH6R

150
MB/s

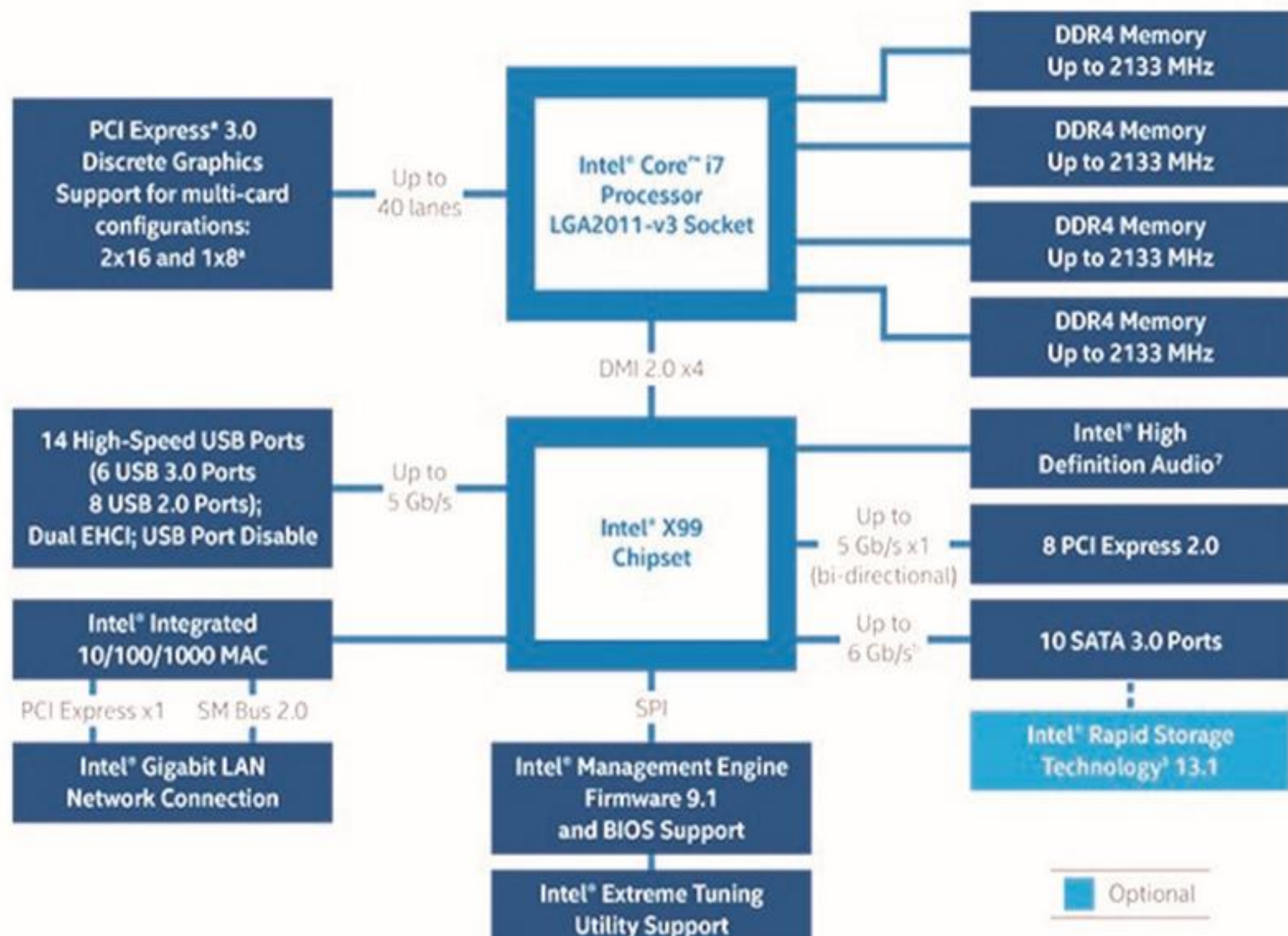
133
MB/s

4 Serial
ATA Ports

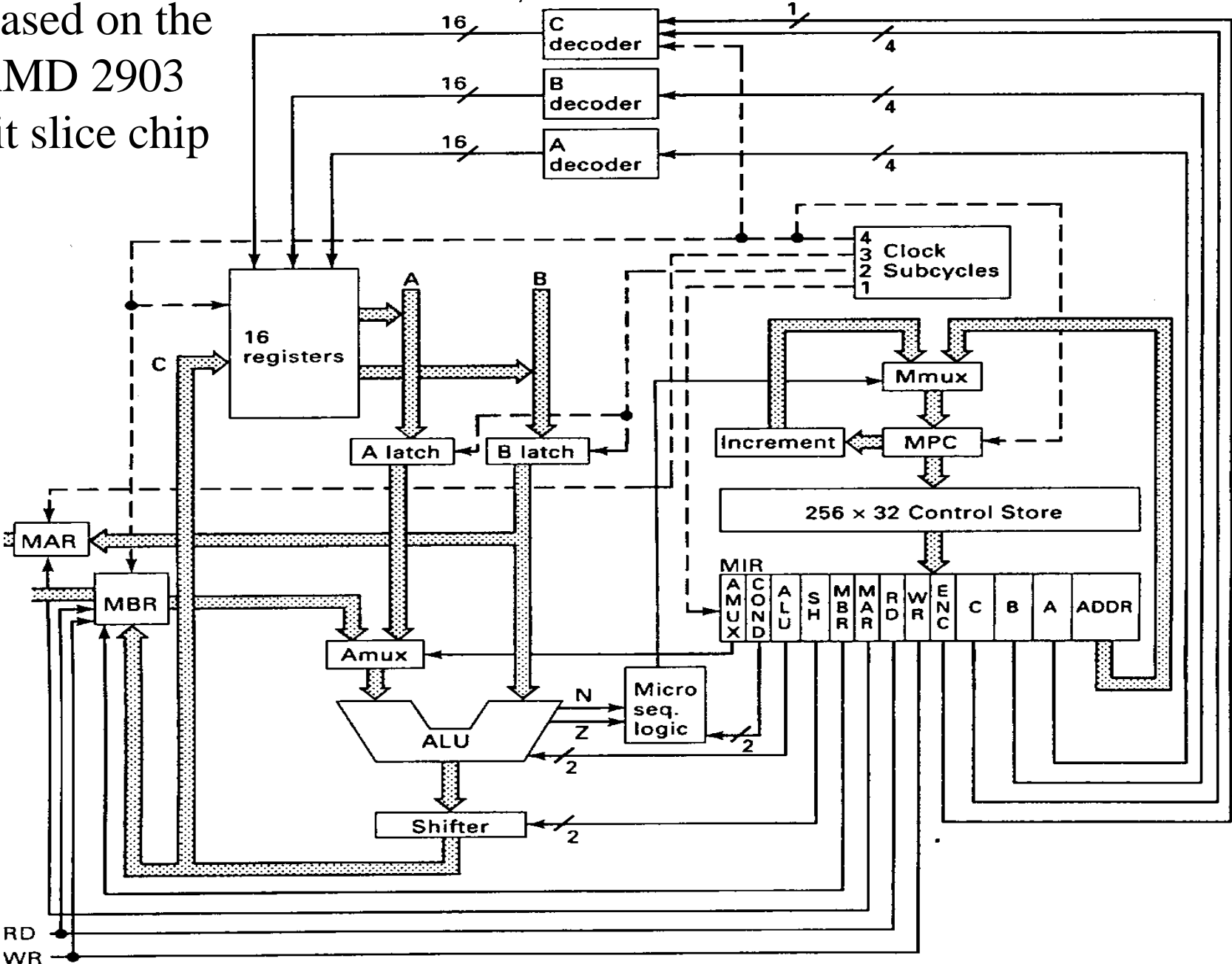
6 PCI

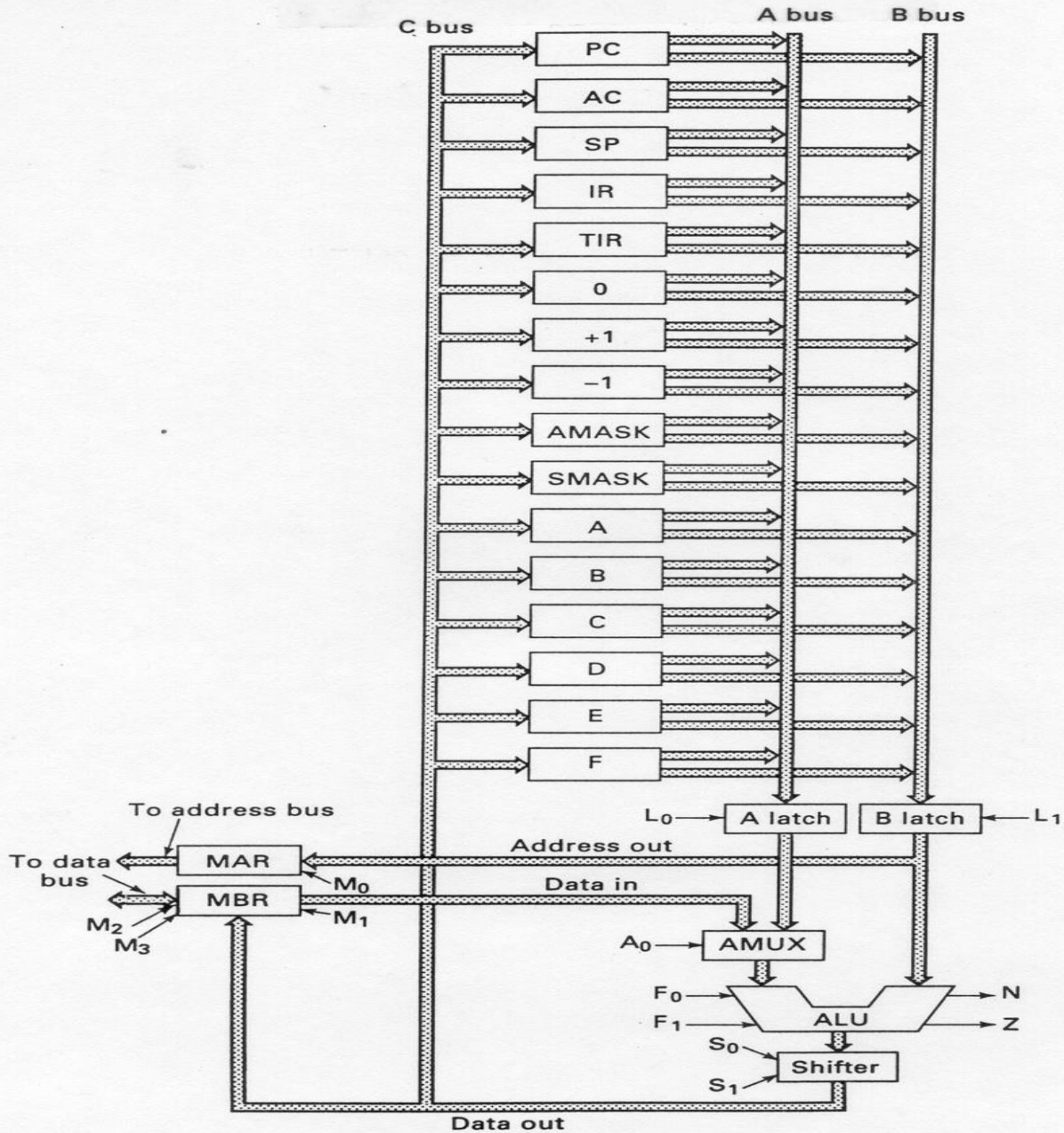
Intel® Matrix
Storage Technology

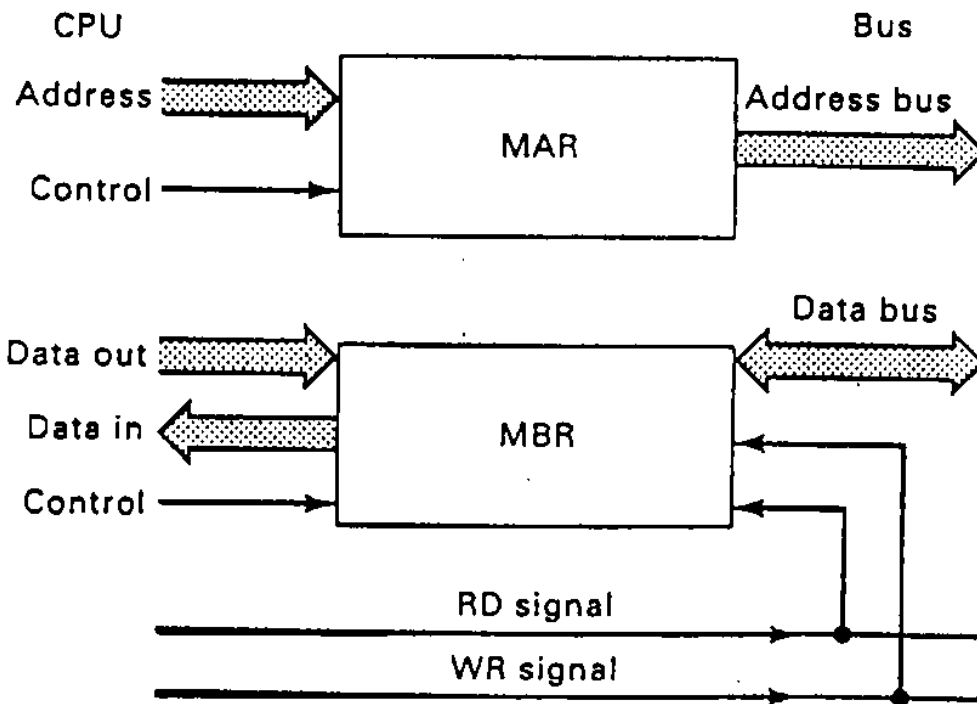
BIOS Supports
HT Technology*



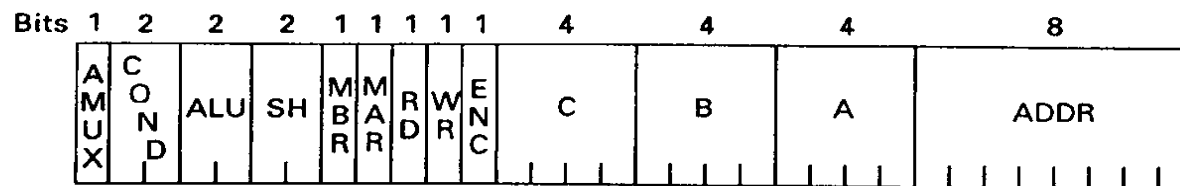
Based on the
AMD 2903
bit slice chip







- AMUX** — controls left ALU input: 0 = A latch, 1 = MBR
ALU — ALU function: 0 = A + B, 1 = A AND B, 2 = A, 3 = \bar{A}
SH — shifter function: 0 = no shift, 1 = right, 2 = left
MBR — loads MBR from shifter: 0 = don't load MBR, 1 = load MBR
MAR — loads MAR from B latch: 0 = don't load MAR, 1 = load MAR
RD — requests memory read: 0 = no read, 1 = load MBR from memory
WR — requests memory write: 0 = no write, 1 = write MBR to memory
ENC — controls storing into scratchpad: 0 = don't store, 1 = store
C — selects register for storing into if ENC = 1: 0 = PC, 1 = AC, etc.
B — selects B bus source: 0 = PC, 1 = AC, etc.
A — selects A bus source: 0 = PC, 1 = AC, etc.



| | | | | |
|------------------------|--------------------------------------------------------------------------|----------------------------------------------------|---------------------------------------------------------------------------------|--------------------------|
| AMUX | COND | ALU | SH | MBR, MAR, RD, WR, |
| 0 = A latch 1 = MBR | 0 = No jump 1 = Jump if N = 1 2 = Jump if Z = 1 3 = Jump always | 0 = A + B 1 = A AND B 2 = A 3 = \bar{A} | 0 = No shift 1 = Shift right 1 bit 2 = Shift left 1 bit 3 = (not used) | 0 = No 1 = Yes |

| | |
|-----------------------------------------------|-------------------------------------------|
| 0:mar := pc; rd; | { main loop } |
| 1:pc := 1 + pc; rd; | { increment pc } |
| 2:ir := mbr; if n then goto 28; | { save, decode mbr } |
| 3:tir := lshift(ir + ir); if n then goto 19; | |
| 4:tir := lshift(tir); if n then goto 11; | { 000x or 001x? } |
| 5:alu := tir; if n then goto 9; | { 0000 or 0001? } |
| 6:mar := ir; rd; | { 0000 = LODD } |
| 7:rd; | |
| 8:ac := mbr; goto 0; | |
| 9:mar := ir; mbr := ac; wr; | { 0001 = STOD } |
| 10:wr; goto 0; | |
| 11:alu := tir; if n then goto 15; | { 0010 or 0011? } |
| 12:mar := ir; rd; | { 0010 = ADDD } |
| 13:rd; | |
| 14:ac := ac + mbr; goto 0; | |
| 15:mar := ir; rd; | { 0011 = SUBD } |
| 16:ac := 1 + ac; rd; | { Note: $x - y = x + 1 + \text{not } y$ } |
| 17:a := inv(mbr); | |
| 18:ac := a + ac; goto 0; | |
| 19:tir := lshift(tir); if n then goto 25; | { 010x or 011x? } |
| 20:alu := tir; if n then goto 23; | { 0100 or 0101? } |
| 21:alu := ac; if n then goto 0; | { 0100 = JPOS } |
| 22:pc := band(ir, amask); goto 0; | { perform the jump } |
| 23:alu := ac; if z then goto 22; | { 0101 = JZER } |
| 24:goto 0; | { jump failed } |
| 25:alu := tir; if n then goto 27; | { 0110 or 0111? } |
| 26:pc := band(ir, amask); goto 0; | { 0110 = JUMP } |
| 27:ac := band(ir, amask); goto 0; | { 0111 = LOCO } |
| 28:tir := lshift(ir + ir); if n then goto 40; | { 10xx or 11xx? } |
| 29:tir := lshift(tir); if n then goto 35; | { 100x or 101x? } |
| 30:alu := tir; if n then goto 33; | { 1000 or 1001? } |

| | |
|-------------------------------------------|------------------------|
| 31:a := sp + ir; | { 1000 = LODL } |
| 32:mar := a; rd; goto 7; | |
| 33:a := sp + ir; | { 1001 = STOL } |
| 34:mar := a; mbr := ac; wr; goto 10; | |
| 35:alu := tir; if n then goto 38; | { 1010 or 1011? } |
| 36:a := sp + ir; | { 1010 = ADDL } |
| 37:mar := a; rd; goto 13; | |
| 38:a := sp + ir; | { 1011 = SUBL } |
| 39:mar := a; rd; goto 16; | |
| 40:tir := lshift(tir); if n then goto 46; | { 110x or 111x? } |
| 41:alu := tir; if n then goto 44; | { 1100 or 1101? } |
| 42:alu := ac; if n then goto 22; | { 1100 = JNEG } |
| 43:goto 0; | |
| 44:alu := ac; if z then goto 0; | { 1101 = JNZE } |
| 45:pc := band(ir, amask); goto 0; | |
| 46:tir := lshift(tir); if n then goto 50; | |
| 47:sp := sp + (-1); | { 1110 = CALL } |
| 48:mar := sp; mbr := pc; wr; | |
| 49:pc := band(ir, amask); wr; goto 0; | |
| 50:tir := lshift(tir); if n then goto 65; | { 1111, examine addr } |
| 51:tir := lshift(tir); if n then goto 59; | |
| 52:alu := tir; if n then goto 56; | |
| 53:mar := ac; rd; | { 1111000 = PSHI } |
| 54:sp := sp + (-1); rd; | |
| 55:mar := sp; wr; goto 10; | |
| 56:mar := sp; sp := sp + 1; rd; | { 1111001 = POPI } |
| 57:rd; | |
| 58:mar := ac; wr; goto 10; | |
| 59:alu := tir; if n then goto 62; | |


```

60:sp := sp + (-1);           { 1111010 = PUSH }
61:mar := sp; mbr := ac; wr; goto 10;
62:mar := sp; sp := sp + 1; rd;   { 1111011 = POP }
63:rd;
64:ac := mbr; goto 0;
65:tir := lshift(tir); if n then goto 73;
66:alu := tir; if n then goto 70;
67:mar := sp; sp := sp + 1; rd;   { 1111100 = RETN }
68:rd;
69:pc := mbr; goto 0;
70:a := ac;                     { 1111101 = SWAP }
71:ac := sp;
72:sp := a; goto 0;
73:alu := tir; if n then goto 76;
74:a := band(ir, smask);         { 1111110 = INSP }
75:sp := sp + a; goto 0;
76:tir := tir + tir; if n then goto 80;
77:a := band(ir, smask);         { 11111110 = DESP }
78:a := inv(a);
79:a := a + 1; goto 75;
80:rd; wr;                       { 11111111 = HALT }

```