Introduction to Z-80 Assembly Language Programming

VCF SE 3.0
Malcolm Macleod
3 May 2015
malcolm@avitech.com.au

```
     = 3C00       00022 TOS     EQU     3C00H          ; Pointer to top-left corner of display
                  00023
                  00024
                  00025         ORG     6000H          ; Point to storage for variables
                  00026
                  00027
6000              00028 COUNTER DS      2              ; Reserve storage for 16-bit variable
6002              00029 STATUS  DS      1              ; Reserve storage for 8-bit variable
                  00030
                  00031
                  00032         ORG     6100H          ; Point to where our program will load
                  00033
                  00034
6100   310080     00035 BEGIN   LD      SP,8000H       ; Initialise the stack pointer
                  00036
6103   21003C     00037 CLS     LD      HL,TOS         ; Point HL at display top-left corner
6106   010004     00038         LD      BC,16*64       ; Number of bytes to clear
6109   1620       00039         LD      D,' '          ; Value to [illegible] ASCII SPACE
610B   72         00040 NEXT    LD      (HL),D         ; Store a SPC at loc'n pointed to by HL
610C   23         00041         INC     HL             ; Increment address pointer
610D   0B         00042         DEC     BC             ; Decrement bytes-to-go counter
610E   78         00043         LD      A,B            ; Put high byte of counter into A
610F   B1         00044         OR      C              ; Logical OR A with low byte of counter
6110   20F9       00045         JR      NZ,NEXT                ; Loop back unless BC = 0
                  00046
6112   211F61     00047 DISPMSG LD      HL,MSG1
6115   11D43D     00048         LD      DE,7*64 + 32-12 + TOS  ; Start at Line 7, Col 20
6118   011800     00049         LD      BC,MSG2-MSG1           ; 24D
611B   EDB0       00050         LDIR
                  00051
611D   18FE       00052 LOOP    JR      $
                  00053
                  00054
611F   57656C63   00055 MSG1    DB      "Welcome to VCF East 9.1!"
       6F 6D 65 20 74 6F 20 56
       43 46 20 45 61 73 74 20
       39 2E 31 21
                  00056 MSG2
                  00057
                  00058         END     BEGIN
```

Topics Covered Today

- Overview of the Z80 – History & Features
- Pinouts and Architecture
- Instruction Set
- Assembly Language Example 1 - Toggle output port
- Development Environment
  - For assembling under Windows
  - For assembling under CP/M 2.2
- Assembly Language Example 2 – Output string to console
  - Assembling Example 2 under Windows
  - Assembling Example 2 in CP/M emulator
- Links:
  - Reference Cards and Manuals
  - Useful Websites

Overview of the Zilog Z-80 CPU

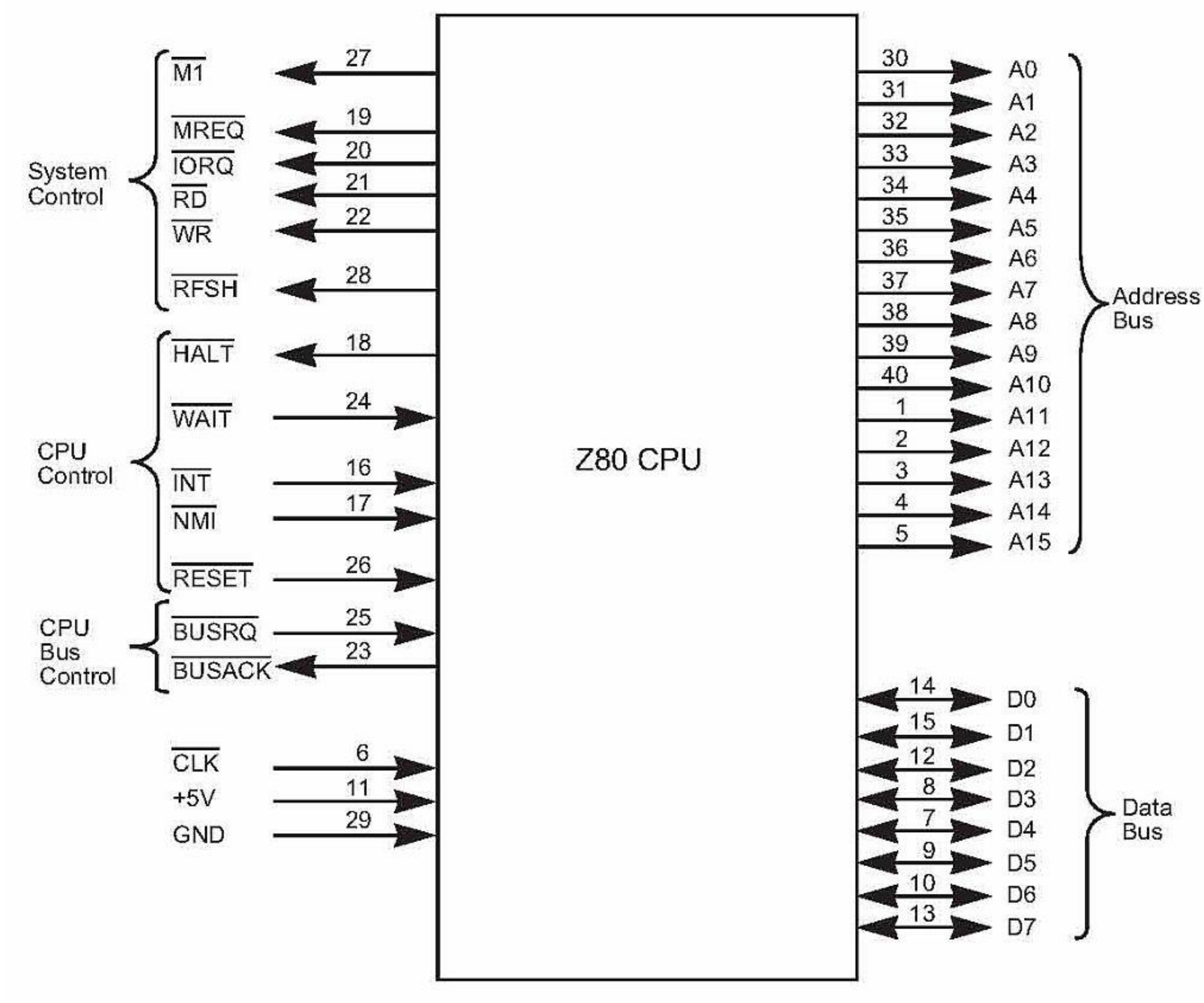
- Released in 1976
- 16-bit address space, 8 bit data bus
- Each instruction stored as 1, 2, 3 or 4 bytes
- “binary upwards compatible” with 8080 machine code – e.g. CP/M 2.2
- Developed by ex-Intel employees: Federico Faggin, Ralph Ungermann and Masatoshi Shima.
- Less than 50% of all Z80 CPUs were produced by Zilog
- Second sourced (licensed) manufactures included: Mostek, Toshiba, Sharp, NEC and SGS-Thomson
- NMOS versions were 2.5 MHz to 8 MHz
- CMOS versions are 4 MHz to 20 MHz

Improvements over the Intel 8080

Relative to the 8080 the Z80 has:

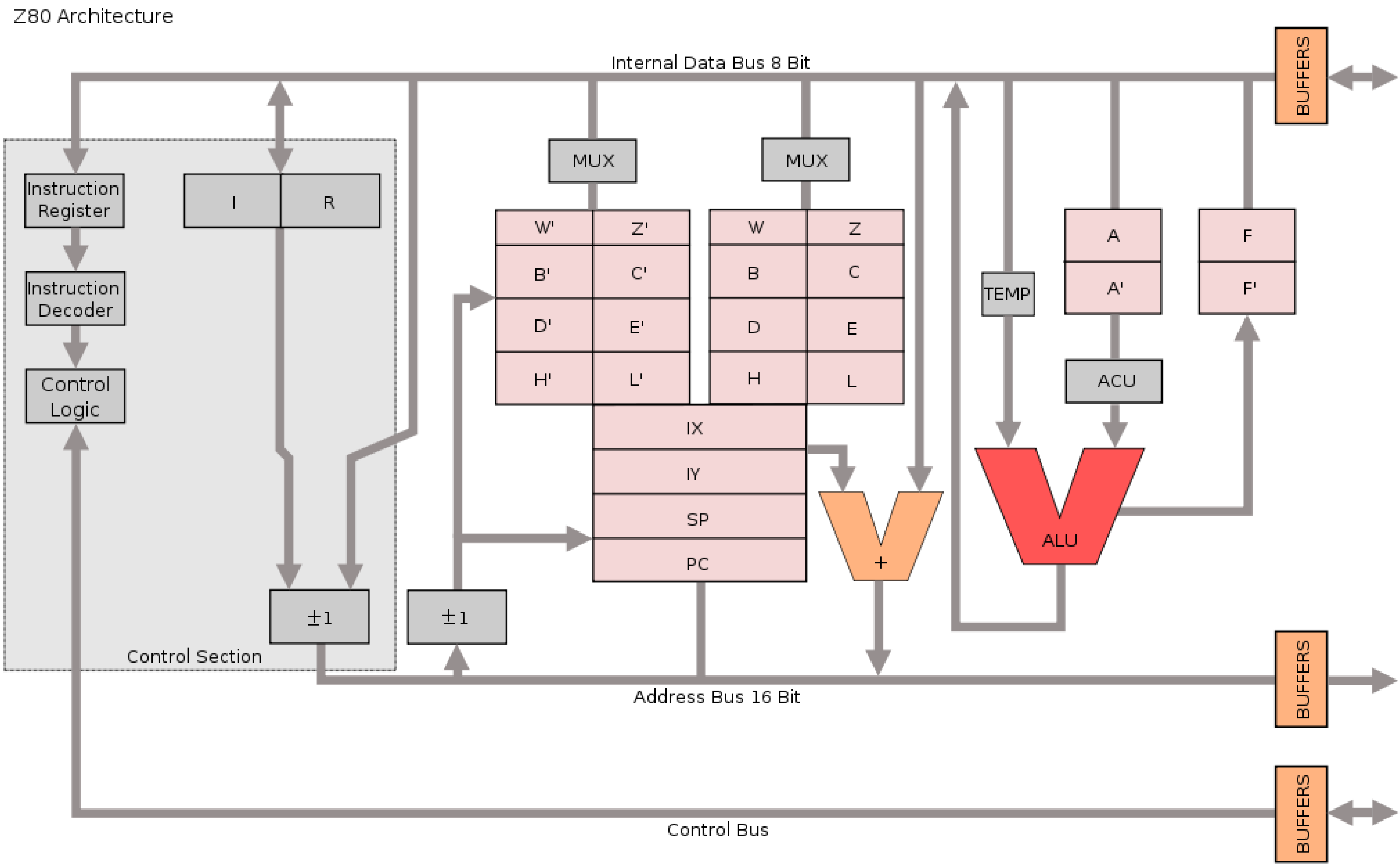
- An enhanced instruction set
- Two new 16-bit index registers (IX & IY)
- 4 new “alternate” 16-bit registers (AF’, BC’, DE’ and HL’)
- Two new interrupt modes (Modes 1 & 2)
- Register I = Interrupt vector base, for Mode 2 interrupts
- Register R = Refresh register
- A non-maskable interrupt input
- Single supply rail (+5V), rather than +5, -5 & +12
- Built in DRAM refresh (only 16k RAMs and smaller)

Z-80 Pin Configuration (40-Pin DIP)



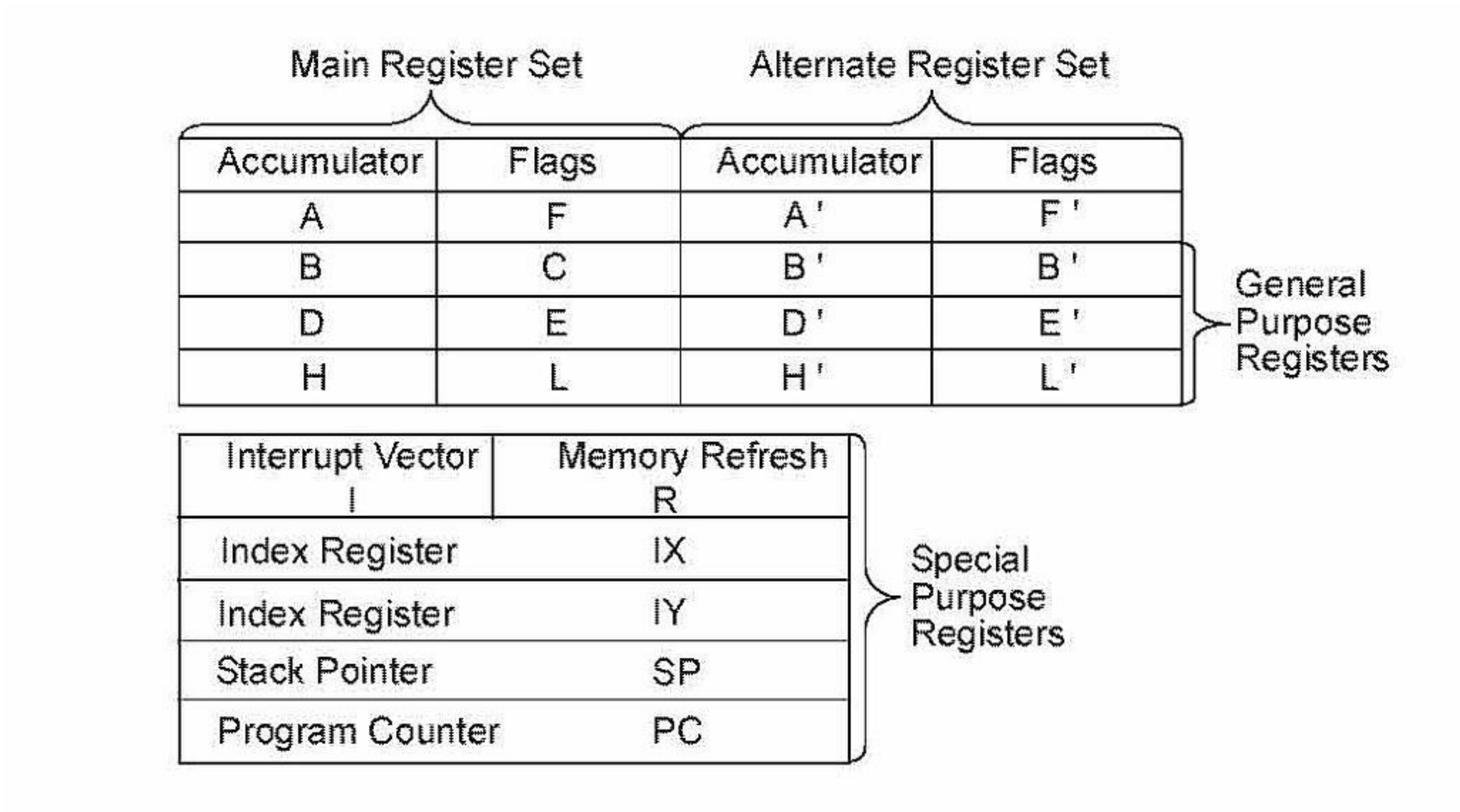
System Control
$\overline{\text{M1}}$ 27
$\overline{\text{MREQ}}$ 19
$\overline{\text{IORQ}}$ 20
$\overline{\text{RD}}$ 21
$\overline{\text{WR}}$ 22
$\overline{\text{RFSH}}$ 28
CPU Control
$\overline{\text{HALT}}$ 18
$\overline{\text{WAIT}}$ 24
$\overline{\text{INT}}$ 16
$\overline{\text{NMI}}$ 17
$\overline{\text{RESET}}$ 26
CPU Bus Control
$\overline{\text{BUSRQ}}$ 25
$\overline{\text{BUSACK}}$ 23
$\overline{\text{CLK}}$ 6
+5V 11
GND 29
Z80 CPU
30 A0
31 A1
32 A2
33 A3
34 A4
35 A5
36 A6
37 A7
38 A8
39 A9
40 A10
1 A11
2 A12
3 A13
4 A14
5 A15
Address Bus
14 D0
15 D1
12 D2
8 D3
7 D4
9 D5
10 D6
13 D7
Data Bus

Architecture



Source: http://en.wikipedia.org/wiki/Zilog_Z80#/media/File:Z80_arch.svg Created by Appaloosa, subject to CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0/). Unmodified.

Registers



Main Register Set		Alternate Register Set	
Accumulator	Flags	Accumulator	Flags
A	F	A'	F'
B	C	B'	B'
D	E	D'	E'
H	L	H'	L'

General Purpose Registers

Interrupt Vector I	Memory Refresh R
Index Register	IX
Index Register	IY
Stack Pointer	SP
Program Counter	PC

Special Purpose Registers

Source: Zilog, Z80 Family CPU User Manual , Document Number UM008002-0202, Figure 2 (Page 3). Unmodified.

Instruction Overview

The Z-80 can execute 158 different (published) instruction types, including all 78 of the 8080A CPU.

The instructions fall into these categories:

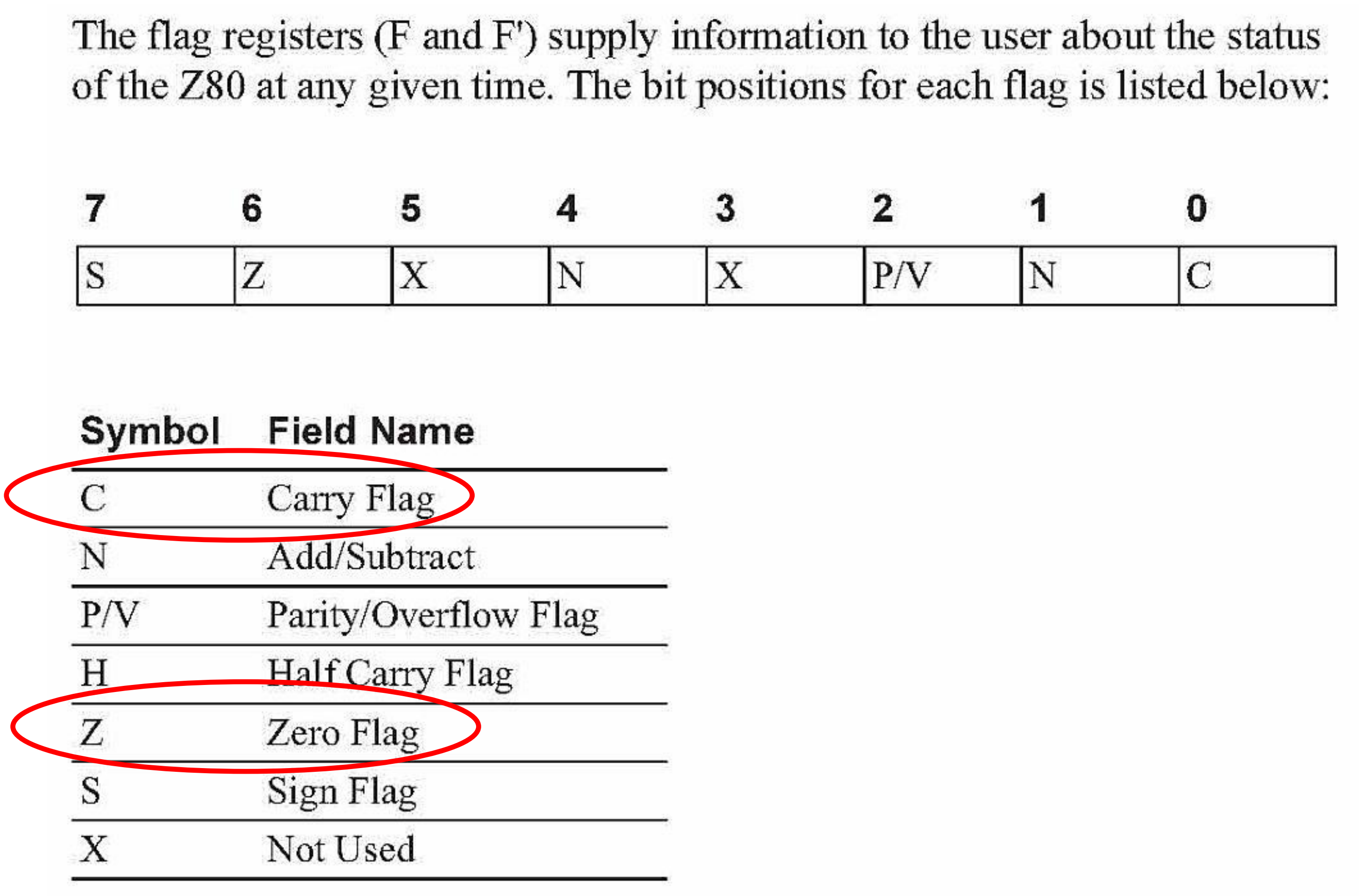
- Load and Exchange
- Block Transfer and Search
- Arithmetic and Logical
- Rotate and Shift
- Bit Manipulation (Set, Reset, Test)
- Jump, Call, and Return
- Input and Output
- Basic CPU Control

Addressing Modes

Most instructions need to access data in external memory or internal CPU registers. The various “addressing modes” describe the way in which this can occur:

Addressing Mode	Assembly Language Example
Immediate	LD A,FFH -or- LD BC,1234H
Modified Page Zero Addressing	RST 30H
Relative Addressing	JR Z,EXIT
Extended Addressing	JP EXIT -or- LD A,(TIMER)
Indexed Addressing	LD A,(IX+9H)
Register Addressing	LD A,B
Implied Addressing	SUB 30H
Register Indirect Addressing	LD A,(HL)

Flags



The flag registers (F and F') supply information to the user about the status of the Z80 at any given time. The bit positions for each flag is listed below:

7	6	5	4	3	2	1	0
S	Z	X	N	X	P/V	N	C

Symbol	Field Name
C	Carry Flag
N	Add/Subtract
P/V	Parity/Overflow Flag
H	Half Carry Flag
Z	Zero Flag
S	Sign Flag
X	Not Used

Notes:

1. When starting out, focus on learning how to use the C and Z flags, then S.
2. Flags H and N cannot be tested – they are only used for BCD arithmetic.

Example of Zilog’s instruction tables

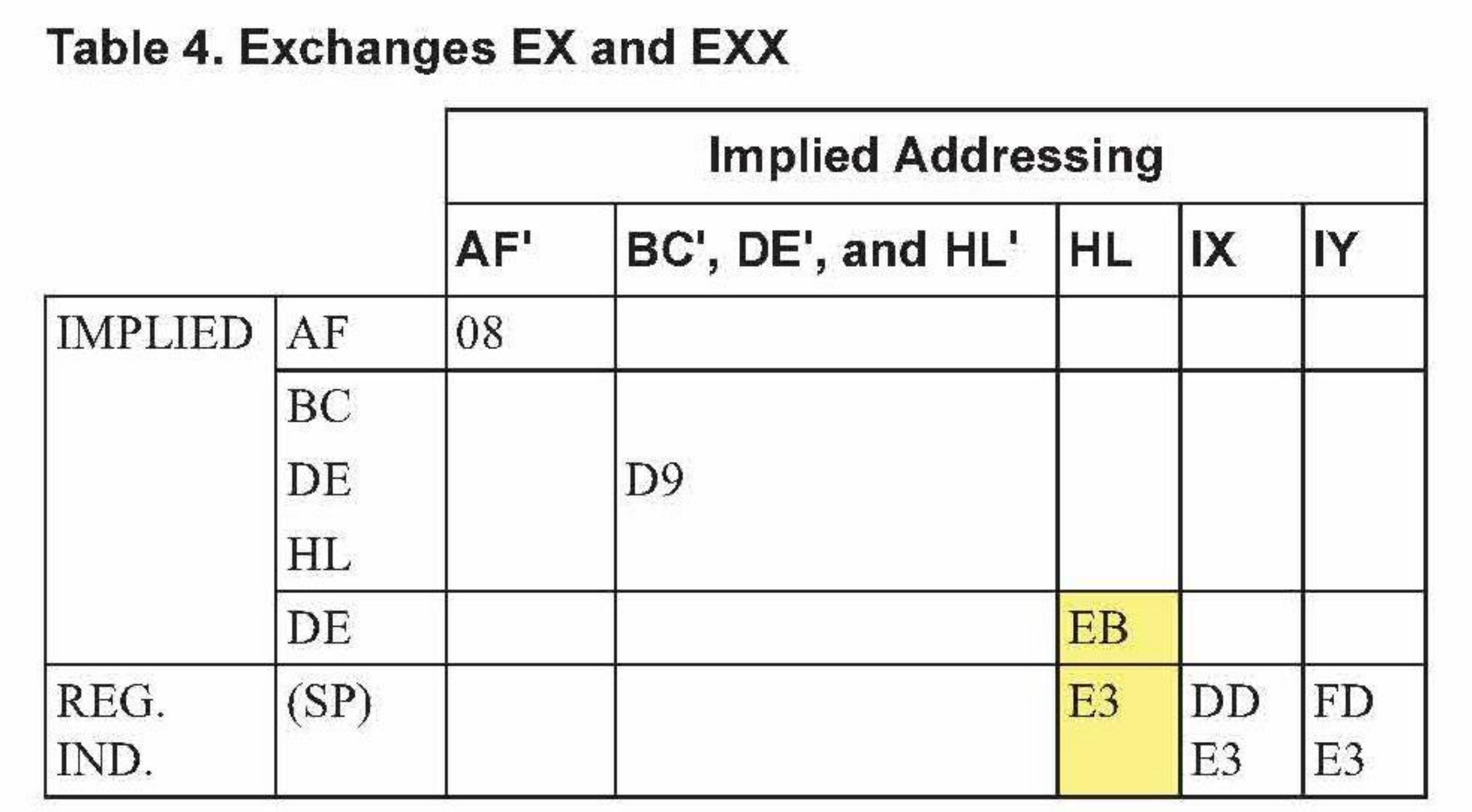


Table 4. Exchanges EX and EXX

		Implied Addressing				
		AF'	BC', DE', and HL'	HL	IX	IY
IMPLIED	AF	08				
	BC DE HL		D9			
	DE			EB		
REG. IND.	(SP)			E3	DD E3	FD E3

Assembly Language Example 1 – Toggle Output

```
CYCLS   EQU    40000D         ; 10 msec = 4,000,000 divided by 100
PORT    EQU    0FFH           ; We're going to toggle output port 0FFH

        ORG    0000H          ; Our program goes in low memory

START   LD     SP,8000H       ; Initialise the stack
LOOP    LD     A,D            ; Get current value of D
        XOR    1H             ; Toggle the least significant bit
        LD     D,A            ; Save new value of D
        OUT    (PORT),A       ; Output new value of D
        CALL   DELAY          ; Delay for 10 msec
        JR     LOOP           ; Loop back to toggle again

DELAY   LD     BC,CYCLS/26D   ; Number of loops required
DELLOOP LD     A,B            ; Put upper 8 bits of BC into A
        OR     C              ; Logical or A with lower 8 bits of BC
        DEC    BC             ; Decrement loop counter
        JR     NZ,DELLOOP     ; Loop unless BC=0
        RET                   ; Return to main program loop

        END    START
```

Assembly Language Example 1 – Toggle Output

```
  -    9C40                 CYCLS  EQU     40000D
  -    00FF                 PORT   EQU     0FFH

  -    0000                        ORG     0000H

 0+10  0000  310080   START  LD      SP,8000H
10+4   0003  7A       LOOP   LD      A,D
14+7   0004  EE01            XOR     1H
21+4   0006  57              LD      D,A
25+11  0007  D3FF            OUT     (PORT),A
36+17  0009  CD0E00          CALL    DELAY
53+12  000C  18F5            JR      LOOP

65+10  000E  010206   DELAY  LD      BC,CYCLS/26D
75+4   0011  78       DELLOOP LD     A,B
79+4   0012  B1              OR      C
83+6   0013  0B              DEC     BC
89+7+5 0014  20FB            JR      NZ,DELLOOP
96+10  0016  C9              RET

  -    0000                  END     START
```

Suggested Windows Environment

Editors:

- Crimson Editor (v. 3.72 – 2008) :

  http://www.crimsoneditor.com/

- Notepad++ (v6.7.5)

  http://notepad-plus-plus.org

Assemblers:

- George Phillips' ZMAC (version 19sep2013):

  http://members.shaw.ca/gp2000/zmac.html

- Matthew Reed's Z80ASM command-line assembler:

  http://www.trs-80emulators.com/z80asm/

Configuring Crimson Editor and ZMAC

Crimson Editor:

- Under *Tools -> Conf. User Tools*, for Hotkey “Ctrl+1”:
  - Set *Menu Text* = zmac
  - Set *Command* = [directory containing zmac]
  - Set *Argument* = $(FileName)
  - Set *Initial Dir* = $(FileDir)
- Use *“.z80”* as suffix for your source code file

ZMAC:

- To assemble, press Ctrl+1 from within Crimson Editor
- Assembled listing will appear as *“.lst”* file in the ./zout directory
- Any assembly errors will also show in the “Capture Output” panel

Suggested Emulated CP/M 2.2 Environment

Editors:

- Crimson Editor
- Notepad++ (v6.7.5)

CP/M Emulator:

- CP/M 2.2 or C/M 3.0 on Peter Schorn's "AltairZ80" SIMH-based emulator

  http://schorn.ch/altair.html

Z80 Assembler:

- SLR Systems' Z80ASM (run this under CP/M)

  http://www.s100computers.com/Software%20Folder/Assembler%20Collection/Assembler%20Collection.htm

Assembling with SLR’s Z80ASM under AltairZ80

1. Download altairz80 from Peter Schorn’s website. The website has versions available for PC, Mac and Linux.
2. Configure a *“cpm2”* file (on your host computer) for altairz80 that attaches *“cpm2.dsk”* and *“i.dsk”* as hdsk0.
3. Create/Edit your *“PROG.Z80”* source file on your host computer.
4. Run altairz80. You’ll get a SIMH *“sim >”* prompt.
5. Type *“do cpm2”* to start CP/M 2.2. [use Ctrl-E later to exit to SIMH]
6. Use *“R.COM”* (on Drive I) to import SLR’s *“Z80ASM.COM”* from your host computer and store it on Drive I.
7. On Drive I, Use *“R PROG.Z80”* to import your source file from the host file system and store it on Drive I.
8. On Drive I, type *“Z80ASM PROG/F”* to assemble your program.
9. On Drive I, type *“W SOURCE.LST”* to export a copy of your *“PROG.LST”* file back to the host file system.

Example 2: Output String to Console

Note: This will be an on-screen demonstration using Crimson Editor, altairz80 and other applications:

1. Assembling under Windows using George Phillips' *zmac* assembler.
2. Assembling under emulated CP/M 2.2 environment using Peter Schorn's *altairz80* emulator and SLR's *Z80ASM* assembler.

Tips & Tricks

- Execution starts at 0x0000
- Remember to initialize SP before doing any calls or push/pop
- Stack grows downwards (and doesn't store at initial value of SP)
- JR can only jump +127/-128. Use JP for longer jumps
- Some instructions do NOT update flags - eg *"LD A,(HL)"*
- Have a strategy about preserving registers - eg *"caller saves"*
- Document your assembly code thoroughly
- There are two interrupt inputs available: /NMI and /INT
- The Z80 is Little Endian (16 bit values are stored LSB first)
- You can assemble to a ROM address, but need to use EPROM programmer to write the program to the chip.
- You can't store variables in ROM!

Key Reference Documents

Z-80 Instant Reference Card:

http://www.ballyalley.com/ml/z80_docs/Z80%20CPU%20Instant%20Reference%20Card%20(Color).pdf

Z-80 Family CPU User Manual

http://www.ballyalley.com/ml/z80_docs/Z80%20Family%20CPU%20User%20Manual%20(Feb%202002)(Zilog)(UM008002).pdf

Rodney Zaks – How to Program the Z80

http://www.ballyalley.com/ml/z80_docs/Programming%20the%20Z-80%203rd%20Edition%20(1980)(Rodnay%20Zaks)(Sybex).pdf

Useful Websites

Documentation:

http://www.ballyalley.com/ml/z80_docs/z80_docs.html

Home of the Z80 CPU – Official Support Page:

http://www.z80.info/

Wikipedia Page on the Z80:

http://en.wikipedia.org/wiki/Zilog_Z80

John Monahan's guide to Peter Schorn's altairz80:

http://www.s100computers.com/Software%20Folder/Altair%20Simmulator/Altair%20Software.htm