# EGCP 2110-01
# Microprocessors
# Laboratory #2

## Addressing Mode

Prepared By:
Alex Laird
Collin Barrett

on

Wednesday, September 9th, 2009

## Source Code

Below is the assembly code that we ensured worked in the lab. We wrote it prior to having access to the Z-80 Microprocessor to test it—this is what we turned in for the pre lab—and after fixing a few syntax errors, we came up with this working code.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Created by: Alex Laird and Collin Barrett
;Date: Sept. 7, 2009
;Class: Microprocessors
;Lab 2: Address Mode
;Purpose: To write an assembly language program that demonstrates
;         the various addressing schemes and logical functions
;         available on the Z-80 microprocessor. To introduce the
;         concept of system ROM subroutine calls for basic input
;         and output.
;Address requirements: Address modes extended, immediate,
;                      register-indirect, implied, indexed, immediate
;                      extended, and register must each be used at
;                      least one time in this program. To show their
;                      use, it will be commented to the right of the
;                      command.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;define some constants needed for the ROM subroutine calls
HEXTO7:     EQU   0FF1H ;calling address for HEXTO7
DISPV:      EQU   1F12H ;calling address for DISPV
KBRD: EQU   0FEBH       ;calling address for KBDR

;clear all the general purpose registers (A-L) to 00H
      LD    A, 00H      ;immediate
      LD    BC, 0000H   ;extended immediate
      LD    D, A        ;register
      LD    E, A
      LD    H, A
      LD    L, A

;store the value FFH into V1
      LD    A, 0FFH
      LD    (V1), A

;store the result of V1 AND 55H into V2
      LD    A, (V1)     ;register-indirect
      AND   55H         ;implied
      LD    (V2), A

;store the result of V1 XOR V2 into V3
      LD    A, (V1)
      LD    B, A
      LD    A, (V2)
      XOR   B
      LD    (V3), A

;store the result of V2 OR 3AH into V4
      LD    A, 3AH
```

```
      LD    B, A
      LD    A, (V2)
      OR    B
      LD    (V4), A

;store the result of V1 XOR V4 into address V5
      LD    A, (V1)
      LD    HL, V4
      XOR   (HL)
      LD    (V5), A

;store 1's complement of V5 into address V6
      LD    A, (V5)
      CPL
      LD    (V6), A

;store the result of V1 XOR V2 XOR V3 XOR V4 XOR V5 XOR V6 into V7
      LD    A, (V1)
      LD    HL, V2
      XOR   (HL)
      LD    HL, V3
      XOR   (HL)
      LD    HL, V4
      XOR   (HL)
      LD    HL, V5
      XOR   (HL)
      LD    HL, V6
      XOR   (HL)
      LD    (V7), A

;rotate the value of V6 four times to the left (without carry) and
store it in V8
      LD    A, (V6)
      RLCA
      RLCA
      RLCA
      RLCA
      LD    (V8), A

;shift V2's value 3 places to the left and store the result into V9
      LD    A, (V2)
      SLA   A
      SLA   A
      SLA   A
      LD    (V9), A

;display the first set and wait for keypress
      LD    IX, DISPV
      LD    A, (V1)
      LD    (IX+0), A    ;indexed
      LD    A, (V2)
      LD    (IX+1), A
      LD    A, (V3)
      LD    (IX+2), A
      CALL  HEXTO7
      CALL  KBRD
```

```
;display the second set and wait for keypress
      LD    A, (V4)
      LD    (DISPV), A
      LD    A, (V5)
      LD    (DISPV+1), A
      LD    A, (V6)
      LD    (DISPV+2), A
      CALL  HEXTO7
      CALL  KBRD

;display the third set and wait for keypress
      LD    A, (V7)
      LD    (DISPV), A
      LD    A, (V8)
      LD    (DISPV+1), A
      LD    A, (V9)
      LD    (DISPV+2), A
      CALL  HEXTO7
      CALL  KBRD

;return control to the ROM-based monitor and return to initial data
entry mode
      JP    0000H       ;extended

;define storage locations
V1:   DEFS  1
V2:   DEFS  1
V3:   DEFS  1
V4:   DEFS  1
V5:   DEFS  1
V6:   DEFS  1
V7:   DEFS  1
V8:   DEFS  1
V9:   DEFS  1
```

## Problems Encountered

The only real problems my partner and I encountered were illegal commands. For instance, numerous times we tried performing an XOR on the accumulator, not realizing that we first had to load the accumulators contents into HL to XOR that. These errors were resolved quickly when we tried to compile five or ten times and quickly fixed them all. Once the syntactical errors were taken care of, our code ran correctly the first time.

## Things Learned

Essentially, the point of the lab was simply to gain experience writing assembly language programs manipulating addressing schemes and logical functions. We were required to use each of the following address modes at least once: extended,

immediate, register-indirect, implied, indexed, extended immediate, and register. All were used and functioned properly in our code.

We learned how to define and use variables, which was really just gaining experiencing in referencing pointers to memory locations and their contents. Additionally, we learned how to call pre-programmed ROM subroutines on the trainer board.

## Conclusions

This lab was a good introduction and first coding assignment for this class, because it was simple and straightforward yet incorporated many basic logical computations that will be useful to know how to use in future assembly coding projects. Furthermore, it was a good, logical introduction to writing programs for the Z-80 microprocessor.