ECE 319 : Digital System Design : Fall 2016

Project II : Due: October 31, 2016.

Design a 8 bit processor with the following specifications[1]:

1. The processor has seven 8-bit registers A, B, C, D, E, H and L.
2. It connects with an external memory of 64 KBytes. The memory has a tristate data output with active low signals *rd* and *wr*.
3. On reset, the processor begins by executing the instruction at address 0.
4. It has instructions MOV, MVI, LXI, LDA, STA, ADD, SUB, NOP and HLT. It can also add or subtract data directly from memory.
5. Within the machine instructions, the registers A, B, C, D, E, H and L are coded as 111, 000, 001, 010, 011, 100 and 101 respectively. Letter M always refers to the contents of the memory address specified by the H-L pair. The machine codes of the chosen instructions are defined in 8085 as follows:

```
01 d2d1d0 s2s1s0 : MOV r1, r2  (copy reg s2s1s0 to register d2d1d0)
01 110 s2s1s0    : MOV M, r    (copy reg s2s1s0 to memory)
01 d2d1d0 110    : MOV r, M    (copy memory contents to register d2d1d0)
01 110 110       : HLT         (halt the processor)
10 000 s2s1s0    : ADD r       (add reg s2s1s0 to register A)
10 000 110       : ADD M       (add memory contents to register A)
10 010 s2s1s0    : SUB r       (subtract reg s2s1s0 from reg A)
10 010 110       : SUB M       (subtract memory contents from register A)
11 000 011       : JMP d16     (jump to address d16)
11 000 010       : JNZ d16     (jump to address d16 if zero flag not set)
11 001 010       : JZ d16      (jump to address d16 if zero flag is set)
00 110 010       : STA d16     (store A to memory address in the next two
                                locations)
00 111 010       : LDA d16     (load A from memory address in the next two
                                locations)
00 d2d1d0 110    : MVI r, d8   (copy the byte in the next memory
                                location to the register d2d1d0)
00 d2d1d0 001    : LXI rr, d16 (copy the bytes in the next two memory
                                addresses to the register pair d2d1d0)
00 000 000       : NOP         (do not do anything)
```

Note that except for the instructions *JMP*, *JNZ*, *JZ*, *STA*, *LDA*, *MVI* and *LXI* above, all other instructions occupy only one byte in the memory. *MVI r, d8* occupies two bytes in the memory. The first of these is the instruction code given above and the next, the 8 bit data. Similarly, *LXI rr, d16* translates to three bytes in memory. The first byte is the instruction code above and the next two bytes provide the data to be moved into the two registers of the specified pair. The address following the instruction code holds the data for the lower register of the pair and the next, the data for the upper register. The 16 bit address specified by *JMP d16*, *JNZ d16*, *JZ d16*, *STA d16* and *LDA d16* sits in two following memory

[1]The architecture specifications and the instruction set/codes is a subset of the Intel 8085 processor

locations. The lower byte of d16 is at the address immediately following the instruction code and the upper byte is in the following location.

In instruction *LXI rr, d16*, the register pair B-C is refered to by using the code for B in the instruction. Similarly codes for D and H are used to refer to the D-E and H-L register pairs[2].

Use the testbench module provided on the coursesite. It already contains a 1KByte ram that is loaded at the begining of the simulation with the machine code in file Proj2.dat. Note that your CPU needs to put out 16 bit addresses. But only lower 10 of these address bits are being used for the ram. Our programs will therefore be limited to 1023 bytes of machine code.

You may create the machine code to test the processor by writing an 8085 assembly program Proj2.asm (using only the instructions implemented in your processor). Running it through the assembler asm85_319 provided on the coursesite will create a file Proj2.dat which is read into the memory on initialization.

For example, the following program generates the ram data shown below.

```
; File Proj2.asm
     org   0
     lxi   h, P1      ; H-L <-- P1 (=20)
     mov   b, m       ; B <-- 30
     mov   c, b       ; C <-- 30
     mvi   a, 6       ; A <-- 6
     add   b          ; A <-- 36
     mov   m, a       ; save 36 to address 20
     nop
     sub   a          ; A <- 0, zero flag set
     lxi   h, P1+2    ; H-L <-- P1+1 (=21)
     jz    P2
     sub   c          ; A <-- -6
P2:  mov   m, a       ; save 0 to address 21
     hlt              ; halt the processor
P1:  db    30, 0ffh
     end
```

File Proj2.dat:

```
21 14 00 46 48 3E 06 80 77 00 97 21 15 00 CA 12
00 91 77 76 1E FF
```

If the processor implementation is correct, this program will write 36 to memory location 20 and 0 to location 21. Registers A, B, C, H and L would have 0, 30, 30, 0 and 21 respectively when the program ends.

Files Proj2.asm and Proj2.dat are placed on the coursesite for your convenience. In addition, the coursesite also has the 8085 assembler (both windows and linux versions) which will create the required data file from your own assembly programs.

Submit a report on your design that includes at the minimum:

- Problem statement
- Design approach

[2] *LXI SP, d16* is also a valid instruction in 8085. But we do not use it here.

- The complete design including datapath sketch and control description.
- Verilog code of the processor, memory and a properly set-up test bench (including good comments)
- Simulation waveforms of your system showing contents of PC, IR and the registers,
- Any additional design enhancements you might have used to improve the speed or to reduce power.

Place all the modules in the same file named proj2.v. Identify yourself clearly in comments at the begining of the file. Upload your files to the coursesite by 11:59 pm, October 31, 2015. Late penalty will be applied to submissions after the due date. (For this purpose, the date stamp on the web submissions is considered as the submission date). Submit your report in class on Nov 2nd.