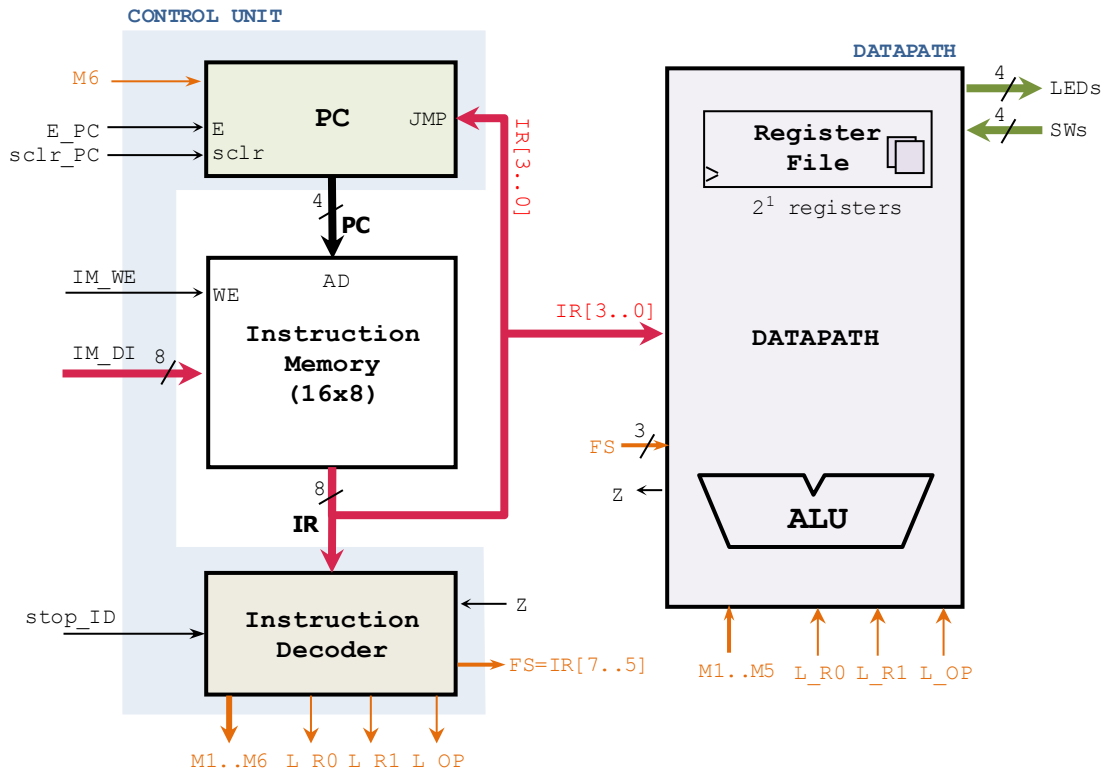# Homework 4

(Due date: `April 11`th `@ 11:59 pm`)

Presentation and clarity are very important! Show your procedure!

PROBLEM 1 (30 PTS)

- **"VBC (Very Basic Computer)"**: 2 registers, 16-word Instruction Memory (IM), 8 bits per instruction (see Notes – Unit 6).

- ✓ Write an assembly program for a counter from 4 to 11: $4, 5, \ldots 11, 4, 5, \ldots$ The count must be shown on the output register (**OUT**). Use labels to specify any address that an instruction may jump to. You can only have up to 16 instructions.

- ✓ Provide the contents of the Instruction Memory. If some instruction bits are unused, you can assign 0's.

Address	Instruction Memory	Assembly Instruction
0000		
0001		
0010		
0011		
0100		
0101		
0110		
0111		
1000		
1001		
1010		
1011		
1100		
1101		
1110		
1111		

✓ Complete the timing diagram for the execution of the previous program. Use unsigned decimal values for R0, R1, OUT, and PC. Specify IR in hexadecimal. (18 pts)

IM: Array of registers. In this case, when reading, output data appears as soon as address is ready.

Assumptions: the program is already in IM. Also: `sclr_PC=0, IM_WE=0, stop_ID=0`.

PROBLEM 2 (18 PTS)

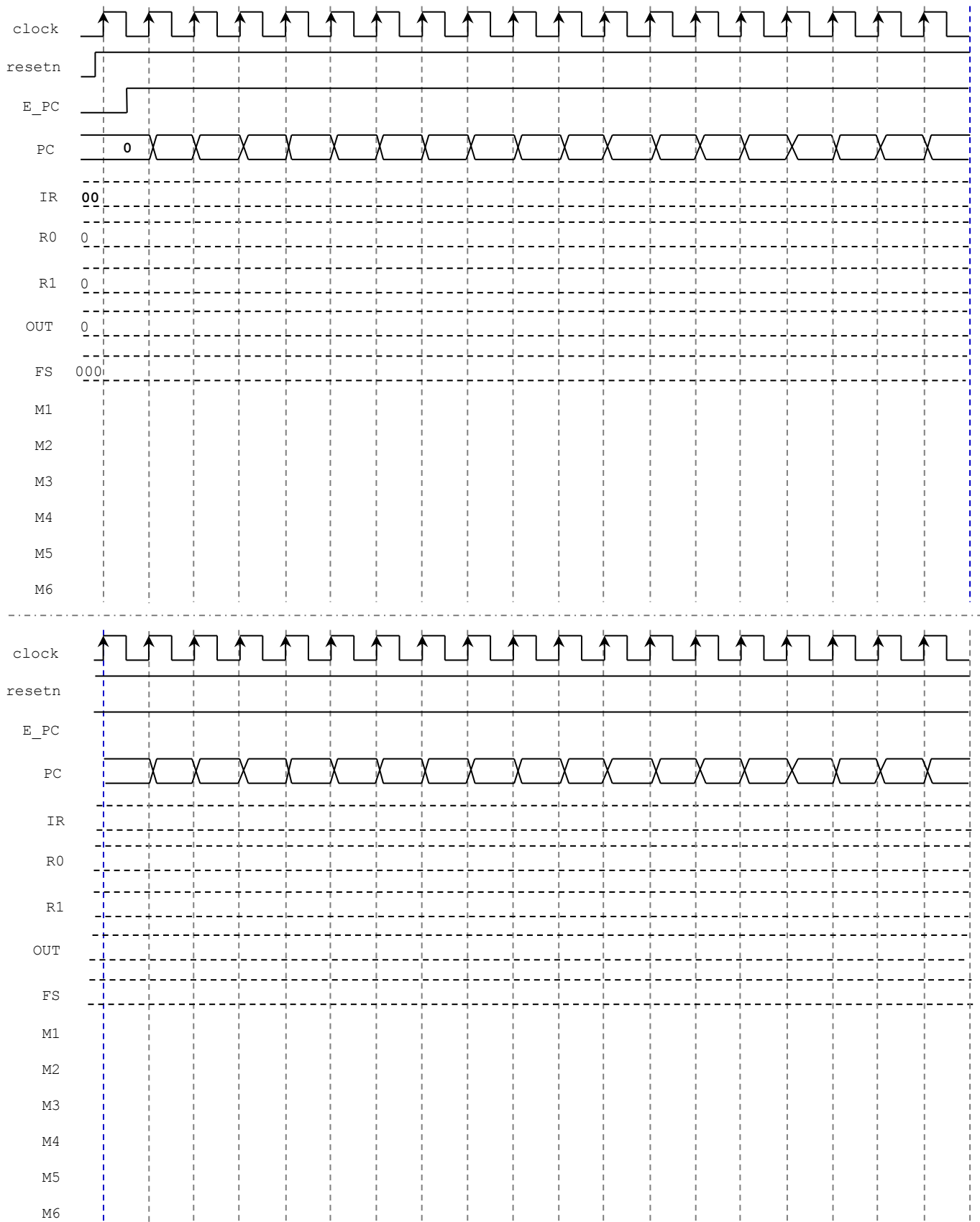- "**Simple Computer**": 8 registers, 64-word Instruction Memory (IM), 16 bits per instruction (see Notes – Unit 6).
  - ✓ For each of the following instructions, indicate the bits for the instruction fields (if they do not apply, write 'NA'), and the instruction memory contents (if some bits are unused, you can assign 0's). (8 pts)

Address	Instruction	OPCODE	DR	SA	SB	OP	AD	Instruction Memory contents
0	LDI R0,5							
1	LDI R3,4							
2	LDI R2,7							
3	SUB R7,R3,R0							
4	ADI R5,R7,7							
5	ST R3,R5							
6	XOR R0,R5,R2							
7	LD R6, R3							
8	BRN R7,-5							
9	SHL R7,R2							
10	JMP R2							

- ✓ Emulate the execution of the program by completing the state of the registers (use hexadecimal values) after the instruction pointed by PC is executed. Then, complete the PC (use binary values) of the next instruction. Complete it until the JMP instruction is reached (execute this instruction as well). (10 pts)

PC	R0	R2	R3	R5	R6	R7
000000						

PROBLEM 3 (17 PTS)

- "**Simple Computer**": 8 registers, 64-word IM, 16 bits per instruction (see Notes - Unit 6)
  - ✓ Write an assembly program that stores numbers from 37 down to 25 in Data Memory (DM) on addresses 0 to 12. Use labels to specify any address that an instruction may jump to.
    - ▫ Note that you can only have up to 64 instructions.
    - ▫ The ALU treats data as signed numbers. Thus, data on the registers (R0 – R7) and Data Memory (16 bits) is signed.
  - ✓ Provide the contents of the Instruction Memory (IM) and its addresses.

address	DM
000000	25
000001	24
000010	23
000011	22
000100	21
000101	20
000110	1F
000111	1E
001000	1D
001001	1C
001010	1B
001011	1A
001100	19
...	

PROBLEM 4 (20 PTS)

- "**PicoBlaze MicroProcessor**": 16 registers, 1024-word Instruction Memory, 18 bits per instruction (see Notes – Unit 6).
  - ✓ Emulate the execution of this program by completing the state of the registers (hexadecimal values) and the status flags (C, Z) after the instruction pointed by PC is executed. Then, complete the PC (hexadecimal values) of the next instruction and update SP (binary values). Complete it until the `jump` instruction (`003`) is reached (execute this instruction as well).

```
Address │ Assembly Program
  000   │ main: load s3,04
  001   │       load s4,05
  002   │       call umult
  003   │       jump main
  ...   │       ...

  067   │ umult: load s5,00
  068   │        load s7,08
  069   │ loop:  sr0 s4
  06A   │        jump nc, sh_p
  06B   │        add s5, s3
  06C   │  sh_p: sra s5
  06D   │        sra s6
  06E   │        sub s7,01
  06F   │        jump nz, loop
  070   │        return
  ...   │        ...
```

Emulation trace (PC = address of instruction executed; register/flag values are the state **after** execution):

PC	SP	s3	s4	s5	s6	s7	C	Z
000	00000	04	--	--	--	--	-	-
001	00000	04	05	--	--	--	-	-
002	00001	04	05	--	--	--	-	-
067	00001	04	05	00	--	--	-	-
068	00001	04	05	00	--	08	-	-
069	00001	04	02	00	--	08	1	0
06A	00001	04	02	00	--	08	1	0
06B	00001	04	02	04	--	08	0	0
06C	00001	04	02	02	--	08	0	0
06D	00001	04	02	02	00	08	0	1
06E	00001	04	02	02	00	07	0	0
06F	00001	04	02	02	00	07	0	0
069	00001	04	01	02	00	07	0	0
06A	00001	04	01	02	00	07	0	0
06C	00001	04	01	01	00	07	0	0
06D	00001	04	01	01	00	07	0	1
06E	00001	04	01	01	00	06	0	0
06F	00001	04	01	01	00	06	0	0
069	00001	04	00	01	00	06	1	1
06A	00001	04	00	01	00	06	1	1
06B	00001	04	00	05	00	06	0	0
06C	00001	04	00	02	00	06	1	0
06D	00001	04	00	02	80	06	0	0
06E	00001	04	00	02	80	05	0	0
06F	00001	04	00	02	80	05	0	0
069	00001	04	00	02	80	05	0	1
06A	00001	04	00	02	80	05	0	1
06C	00001	04	00	01	80	05	0	0
06D	00001	04	00	01	40	05	0	0
06E	00001	04	00	01	40	04	0	0
06F	00001	04	00	01	40	04	0	0
069	00001	04	00	01	40	04	0	1
06A	00001	04	00	01	40	04	0	1
06C	00001	04	00	00	40	04	1	1
06D	00001	04	00	00	A0	04	0	0
06E	00001	04	00	00	A0	03	0	0
06F	00001	04	00	00	A0	03	0	0
069	00001	04	00	00	A0	03	0	1
06A	00001	04	00	00	A0	03	0	1
06C	00001	04	00	00	A0	03	0	1
06D	00001	04	00	00	50	03	0	0
06E	00001	04	00	00	50	02	0	0
06F	00001	04	00	00	50	02	0	0
069	00001	04	00	00	50	02	0	1
06A	00001	04	00	00	50	02	0	1
06C	00001	04	00	00	50	02	0	1
06D	00001	04	00	00	28	02	0	0
06E	00001	04	00	00	28	01	0	0
06F	00001	04	00	00	28	01	0	0
069	00001	04	00	00	28	01	0	1
06A	00001	04	00	00	28	01	0	1
06C	00001	04	00	00	28	01	0	1
06D	00001	04	00	00	14	01	0	0
06E	00001	04	00	00	14	00	0	1
06F	00001	04	00	00	14	00	0	1
070	00000	04	00	00	14	00	0	1
003	00000	04	00	00	14	00	0	1

Result: s3 (04) × s4 (05) = 14h (= 20) stored in s6.

PROBLEM 5 (15 PTS)

- Attach a printout of your Project Status Report (no more than 3 pages, single-spaced, 2 columns). This report should contain the current status of the project, including more details about the design and its components. You **<u>MUST</u>** use the provided template (`Final Project – Report Template.docx`).