EG 201- Computer architecture- Endterm exam. Dec 2018
Total - 40 marks, 3 hours

Ques 1 - 8085 memory interfacing: (4+2 marks)
How would you interface a 32kb EPROM and a 32kb RAM with an 8085 processor (Assume each location stores one byte)
Show a block diagram.
What would be the address ranges of the EPROM and RAM?

Soln -
2 marks:
32kb memory requires 15 address lines. So, A14-A0 of 8085 are connected to 15 address pins of both EPROM and RAM.

2 marks:
A15 = 0 will select EPROM,
A15=1 will select RAM. Use an inverter to implement this.

2 marks:
Address range of
EPROM - 0000H to 7FFFH (0 to 2^15 - 1)
RAM- 8000H to FFFFH. (2^15 to 2^16 - 1)

If you consider the memory size as 32kbits, then it has 4k locations each. So, A11-A0 to address each chip. A12 to select between the 2 chips.
Range will be:
0000 - 0FFF for the EPROM (0 to 2^12 - 1)
1000 - 1FFF for the RAM

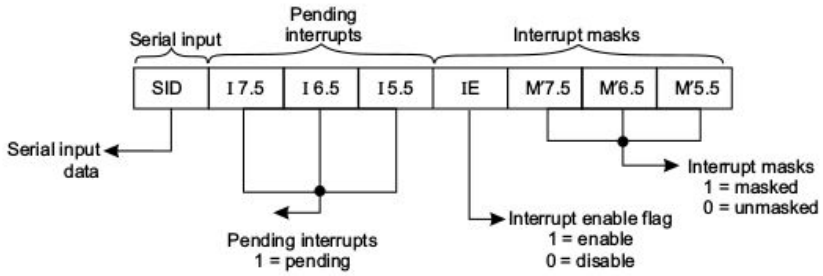
Ques 2 - 8085 (3+3 marks)

a. **Write an 8085 assembly language program to exchange the content of flag register with register C using PUSH and POP instructions.**

```
PUSH PSW    Push value of accumulator and flag in stack
PUSB B
POP PSW
POP B1
```

b. **Write an 8085 program which does the following: It should call the interrupt service routine at 0034 H corresponding to RST 6.5 if it is pending. (Assume that the initial accumulator content that gets read is 10H)**

Soln-



0001 0000 -- 10H - accumulator content
AND this with 0010 0000 - 20H to check if 6.5 is pending

RIM - reads in 10H to accumulator
ANI 20H - to check if 6.5 is pending→ returns a zero if 6.5 is not pending
CNZ 0034H -- calls subroutine if 6.5 was pending. In this case, since accumulator was 10H, it does not call

---

Ques 3 (CPI):

Suppose you have a computer running a program at 2GHz with the following loop running 10 times

```
Loop:  LW $r3, 12 ($r2)
       Add $r6, $r2, $10
       SW $r3, 12 ($r12)
       Add $r6, $r2, $10
       BNE $r6, $r2, loop
```

An ALU operation takes 1 clock cycle, Load 2 cycles, Store 2 cycles and Branch 3 cycles.

(a) What is the CPI? **(2 marks)**

(b) Suppose, we were to introduce an instruction called ALUB which is a combination of

(i) LW + Add (that is, LW followed by ADD) Or

(ii) SW + Add (that is, SW followed by ADD)

Let us replace all such combination of instructions by the new instruction called ALUB. The new instruction takes 3 clock cycle to complete. CPI for the other instructions remains same as before. Write the sequence of the new instruction mix. Compute the CPI for this new version and comment on whether there is a speedup with this ALUB. **(4 marks)**

Soln:

(a) CPU clock cycles = I * CPI = 10*2 + 10*1 + 10* 2 + 10* 1 + 10 * 3 = 90 cycles. -- 2 marks

CPI = CPU clock cycles / Instr = 90/50 = **1.8**

(b) New instr mix = ALUB, ALUB, BNE -- 1 mark
CPU clock cycles = I * CPI = 10*3 + 10*3 + 10 * 3 = 90 cycles. -- 2 marks
CPI = CPU clock cycles / Instr = 90/30 **= 3**. No speedup -- 1 mark

---

Ques 4 - pipelining:

Consider the following instruction sequence

ADD R1, R2,R1
LW R2, 0(R1)
LW R1, 4(R1)
OR R3, R1,R2

Assume that before any of the above instruction is executed, all values in data memory are zeroes and that registers R0 through R3 have the following initial values:
R0 = 10, R1= 1, R2= 31, R3= 1500. Which register value is the first one to be forwarded and what is the value it overrides? **(2 marks)**

Soln:

(R1) Inst1 to Inst2 (32 overwrites 1)

Ques 5 - Pipelining hazards (8 marks)

Consider the following pipeline with 8 stages for the MIPS processor:

IF1 Instruction fetch starts
IF2 Instruction fetch completes
ID Instruction decode and register fetch; begin computing branch target
EX1 Execution starts; branch condition tested; finish computing branch target
EX2 Execution completes - effective address or ALU result available
MEM1/ALUWB First part of memory cycle plus WB of ALU operation
MEM2 Memory access completes
LWB Write back for a load instruction

Consider the following instruction sequence:

```
Loop: LW R1, 0(R1)
      AND R1, R1, R2
      LW R1, 0(R1)
      BEQ R1, R0, Loop
      ADD R10, R11, R5
```

Assume data forwarding and always-taken branch predictor. Write the pipeline diagram for this instruction sequence if in reality the branch is not taken. Indicate stalls/flushes if any.
In which clock cycle does the ADD instruction start fetching?

Soln -
Arrows indicate forwarding.
For load- Forwarding is done for R1 from MEM2 to EX1.
For AND- Forwarding is done for R1 from EX2 to EX1.
Branch target computed in EX1→ so flush 3 instructions if mispredicted
ADD starts fetching in the 15th clock cycle

The first 2 forwarding with stalls - 2 marks each → 4 marks
3rd forwarding- 1 mark
3 flushes -- 2 mark
ADD in 15th clock cycle - 1 mark

Q: 5 a) Write the pipeline diagram for this instruction sequence if in reality the branch is not taken. Indicate stalls/flushes if any.
M: / 7.00

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LW R1, 0(R1)	IF1	IF2	ID	EX 1	EX2	Me m1	Me m2	LW B							
AND R1, R1, R2		sta ll	stall	sta ll	IF1	IF2	ID	Ex1	EX2	Me m1 or AL UW B	Me m2	LW B			
LW R1, 0(R1)						stall	IF1	IF2	ID	Ex1	EX2	Me m1 or AL UW B	Me m2	LW B	
BEQ R1, R0, Loop								stall	stall	stall	IF1	IF2	ID	Ex1	
I-1												flus h			
I-2												flus h			
I-3												flus h			
ADD															IF1

Ques 6 (Caches)-

Consider a direct mapped cache shown in figure below which uses 4-word blocks, 32-bit data words and 32-bit address.

Address

Tag	Index	Offset

Direct mapped cache

	V	Tag	Data0	Data1	Data2	Data3
0	1	0x000004	0x412C1232	0x41111234	0x11241239	0x412C1111
1	1	0xCC3003	0x002C12AB	0x00111235	0x00011238	0x412C1000
2	0	0xAA4004	0x41AC1200	0x41100237	0x115411255	0x412C1133
3	1	0x500001	0x002AAA38	0x001112AA	0x00011222	0x412C1000
4	1	0xA57000	0xAAAC1232	0x41111234	0x11111230	0x412C1BB
5	1	0x000200	0xCA1C1238	0xCC111235	0x030100234	0x412C10A4
15						
			32	32	32	32

1A. Say you are trying to read data from memory location 0x00020013. Assume you are reading the bytes individually. What would be the: **(4 marks)**

a. Tag field value
b. Offset value
c. The cache line or block in which the data should be found
d. Is it a cache hit? If yes, what is the data that is read assuming a byte read?

1B. Can data from locations 0x00012329 and 0x00322FF9 be present in the cache at the same time? Explain. **(2 marks)**

1C. What is the total size of the above cache? **(2 marks)**

1D: Suppose you convert this direct mapped cache into an 8 way set associative cache. Draw the structure of the cache. How many words are present in each way? **(2 marks)** If you get an address 0x00012323. In which cache line or block will you find the data? **(2 marks)**

Solutions:

1A. Assume that you are trying to read data from memory location 0x00020013 and it is a Cache hit. What would be the:

Soln: Address: 0000 0000 0000 0010 0000 0000 **0001** 0011 (32 bit)

16 lines → 4 index bits

Each block or line has 4 words i.e., 16 bytes. To address each byte, we need 4 bits of offset

Remaining 24 bits are tag bits.

(4 marks)

a. Tag field value → 0x000200
b. Block offset value →3 or 0011
c. The cache line in which the data will be found →Index 0001 -- Line 1
d. Data that is read assuming a byte read? →Cache miss.
   Assuming cache hit --AB

1B. Can data from locations 0x00012329 and 0x00322FF9 be present in the cache at the same time? Explain. **(2 marks)**

Soln: Index for 0x00012329 is 2. Index for 0x00322FF9 is F.

Line 2 and F are different indices and index to different cache lines. So, they can be present.

1C. What is the total size of the above cache? **(2 marks)**

Soln: Valid and tag bits are usually ignored while calculating the size - since we are interested in calculating the amount of data that is stored.

16 lines * 4 words * 4 bytes = 256 bytes = 2048 bits

If the tag and valid bits are considered: 16 * (1 + 24 + 16*8) = 2448 bits → Partial marks for this answer.

1D: Suppose you convert this direct mapped cache into an 8 way set associative cache. Draw the structure of the cache. How many words are present in each way? **(2 marks)**
If you get an address 0x00012323. In which cache line or block will you find the data? **(2 marks)**

Soln: 2 rows and 8 ways.

Each way and each line will contain 4 words.

Line 1	4 words	4 words	etc					
Line 2	4 words	4 words						

Last 8 bits of address: 001**0** 0011.
0011- 4 bits to do byte access within each way (4 words or 16 bytes per way)
One bit is needed for index bit. So, 0 is the index. Line 1 is where you will find data.