MINOR II

(EEL308) Computer Architecture

Time: 1 Hour | 2009 | Max. Marks: 22

NAME: Entry No.

N. B.: Do the **rough** calculations on continuation sheet provided. This is **open book / notes examination**, but **transfer** of notes to each other is strictly **prohibited**.

Take the value of *m n* = for this question paper.

Q1 (a):-The values assigned to various registers (In 8086) are:

DS = 2 *m n* 0 H,	DI = 10C0 H,	SI = 03F0 H,	DX = 02D0 H,
SS = 10 *m n* H,	SP = 30 *m n* H,	BP = 20 *m n* H,	BX = 0010 H

Calculate and write the **physical address** of **destination operand** for the instructions below: **(4)**

i) MOV [BX + 0FFFF H], AL
Physical Address = ... H

ii) MOV [BX + SI – 3F H], AL
Physical Address = ... H

iii) MOV [BP+ DI + 7E H], AL
Physical Address =..... H

(b): - Consider two processors, A and B, that have the same basic instruction set, but processor B has a new complex instruction that was found to be able to replace two simpler instructions without increasing the CPI . However it leads to a lower clock frequency. Processor B can execute in 350 MHz while processor A could run in 400 MHz.

How large fraction of the code running on processor A must be replaced by the new instruction in order to reach the same performance for processor B with the lower clock frequency? (2)

Calculations:

Answer: Fraction of the original instructions replaced = %

Q2: - Suppose we are using score-boarding to dynamically schedule code. We have six functional units with the following execution stage latencies.

Functional units	Latency (in cycles):
1: ALU Integer	1
2: FP add/subtract	10
3: FP add/ Subtract	10
4: FP multiplication	5
5: FP multiplication	5
6: FP division	25

Consider the following program.

I. No.	Instruction
1.	I_1: ADDF F_1, F_3, F_2
2.	I_2: SUBF F_5, F_1, F_4
3.	I_3: MULTF F_4, F_6, F_7
4.	I_4: SUBF F_3, F_1, F_2
5.	I_5: DIVF F_1, F_2, F_5

At the start of the program, no register values are being computed by any functional units (i.e., all registers are "ready"). Suppose the first instruction (I_1) issues on cycle 2 and reads operands on cycle 3.

For each of the remaining instructions (**2, 3, 4, 5**), determine whether the instruction can be issued and whether it can read operands within the next five cycles (i.e., by the end of **cycle 8**). If **no**, explain **why not** (in less than 10 words). **Encircle** the **correct option** and explain if answer is in negation. **(4)**

a. Can 2 be issued? Yes / No
If no, why it is not?

b. Can 2 read operands? Yes / No
If no, why it is not?

c. Can 3 be issued? Yes / No
If no, why it is not?

d. Can 3 read operands? Yes / No
If no, why it is not?

e. Can 4 be issued? Yes / No
If no, why it is not?

f. Can 4 read operands? Yes / No
If no, why it is not?

g. Can 5 be issued? Yes / No
If no, why it is not?

h. Can 5 read operands? Yes / No
If no, why it is not?

Q3: - Strictly in **less than 20 words** indicate whether statements are **true** or **false**. (If the statement is **false**, give **reason** why it is false.) **(3)**

(I)– MIPS rating is a poor measure of performance because it completely ignores the effects of clock cycle time.

Answer: True / False (Encircle the correct option)

(IF False give reason):

(II)– Increased pipelining improves clock frequency but does not change IPC (instruction per cycle)
Answer: **True / False** **(Encircle the correct option)**
(IF False give reason):

Q4: - Consider the following sequence of instructions (the machine language used is similar to 8085). The instructions are numbers for easier exposition.

ADD	**R_3, R_1**	**(1)**
SUB	**R_2, R_1**	**(2)**
LOAD	**R_3, 2017**	**(3)**
MUL	**R_2, R_3**	**(4)**
STO	**2050, R_2**	**(5)**

Assume we are implementing this machine language on a **5**-stage pipelined machine and no forwarding latch is implemented in the **ALU**. Assume that the registers are written in the **5th** stage of execution, the memory is read and written in the **4th** stage of execution, and assume that the registers are read in the **2nd** state of execution while loading the latches for the execute cycle. Assume that memory stalls the pipeline until the data returns. **(4)**

(a) Identify (And fill in the table) if any RAW data hazards.

Answer:

Existence of Hazard between Instruction Number	On

(b) Can any **WAR** of **WAW** data hazards occur in this code segment? If so, identify them. If not, state why they cannot occur (Strictly in less than **30 words**).

Answer:

(c) Assume that a forwarding latch is implemented in the ALU. How does that affect the hazards identified in part (a)? (Answer in less than **25 words**)

Answer:

Q3: The following assembly code is to be executed in a three-stage pipelined processor with hazard detection and resolution in each stage. The stages are instruction fetch, operand fetch (one or more as required), and execution (including a write-back operation). Explain all possible hazards in the execution of the code. Try to Schedule it by inserting time delays to avoid hazards. **(5)**

I_1	Inc	R0	/ R0 ← (R0) + 1 /
I_2	Mul	Acc, R0	/ Acc ← (Acc) X (R0)
I_3	Store	R1, Acc	/ R1 ← (Acc) /
I_4	Add	Acc, R0	/ Acc ← (Acc) + (R0) /
I_5	Store	M, Acc	/ M ← (Acc) /

Answer: For understanding existing hazards fill the following appropriately.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
IF									
OF									
EX									

List the existing hazards in the following table.

At T =	Between instruction No.	On	Hazard type

Scheduling which avoids hazard conditions: (Fill the table appropriately without hazards)

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
IF										
OF										
EX										