King Fahd University of Petroleum and Minerals
College of Computer Science and Engineering
Computer Engineering Department

COE 301 COMPUTER ORGANIZATION
ICS 233: COMPUTER ARCHITECTURE & ASSEMBLY LANGUAGE

Term 161 (Fall 2016-2017)
Final Exam
Monday Jan. 16, 2017
12:30-3:00 PM

Time: 150 minutes, Total Pages: 11

Name: _KEY________________________ ID:___________________ Section: _______

Notes:

- Do not open the exam book until instructed
- Answer all questions
- All steps must be shown
- Any assumptions made must be clearly stated
- Mobile phones must be switched off

Question	Max Points	Score
Q1	18	
Q2	20	
Q3	15	
Q4	22	
Total	75	

Dr. Aiman El-Maleh
Dr. Marwan Abu-Amara

[18 Points]

(Q1)

(i) Given two different computers, A and B, the following measurements have been made for running a Java program (Program 1) on these two computers:

Program 1	A	B
#Instructions Executed	5×10^9	8×10^9
CPI	1.2	1.5

Compute the clock rate ratio of computer A to computer B so that the execution time for Program 1 on both computers is the same. **(3 points)**

Execution time for Program 1 on A = $5 \times 10^9 \times 1.2 \times 1/CR_A = (6 \times 10^9)/CR_A$
Execution time for Program 1 on B = $8 \times 10^9 \times 1.5 \times 1/CR_B = (12 \times 10^9)/CR_B$
Since execution time for Program 1 on A = execution time for Program 1 on B
$\Rightarrow (6 \times 10^9)/CR_A = (12 \times 10^9)/CR_B \Rightarrow CR_A/CR_B = (6 \times 10^9)/(12 \times 10^9) = 0.5$

(ii) Consider two different implementations, M1 and M2, of the same instruction set. There are three classes of instructions (A, B, and C) in the instruction set. M1 has a clock rate of 4 GHz and M2 has a clock rate of 2 GHz. The CPI for each instruction class on M1 and M2 is given in the following table:

Class	CPI on M1	CPI on M2	C1 Usage	C2 Usage
A	1	2	40%	30%
B	3	2	50%	30%
C	4	2	10%	40%

The table above also contains a summary of the usage of the instruction classes generated by two different compilers, C1 and C2. Assume for any given program that compiler C1 generates <u>10% more instructions</u> than compiler C2. Which implementation and compiler combination gives the best performance? **(4 points)**

Exec. time for M1 using C1 = $1.1 \times I \times (1 \times 0.4 + 3 \times 0.5 + 4 \times 0.1) \times 1/(4 \times 10^9) = (0.6325 \times I)$ ns
Exec. time for M2 using C1 = $1.1 \times I \times (2 \times 0.4 + 2 \times 0.5 + 2 \times 0.1) \times 1/(2 \times 10^9) = (1.1 \times I)$ ns
Exec. time for M1 using C2 = $I \times (1 \times 0.3 + 3 \times 0.3 + 4 \times 0.4) \times 1/(4 \times 10^9) = (0.7 \times I)$ ns
Exec. time for M2 using C2 = $I \times (2 \times 0.3 + 2 \times 0.3 + 2 \times 0.4) \times 1/(2 \times 10^9) = (1.0 \times I)$ ns

$\Rightarrow$ Implementation M1 and Compiler C1 will give the best performance.

(iii) Compare the performance of a **single-cycle processor** and a **multi-cycle processor**. The delay times are as follows:

Instruction memory access time = 500 ps
Data memory access time = 500 ps
Instruction Decode and Register read = 300 ps
Register write = 100 ps
ALU delay = 300 ps

Ignore the other delays in the multiplexers, wires, etc. Assume the following instruction mix: 40% ALU, 20% load, 10% store, 20% branch, and 10% jump.

a) Compute the delay for each instruction class and the clock cycle for the **single-cycle** processor. **(4 points)**

Instruction Class	Instruction Memory	Decode and Register Read	ALU	Data Memory	Write Back	Total Delay
ALU	500 ps	300 ps	300 ps		100 ps	1200 ps
Load	500 ps	300 ps	300 ps	500 ps	100 ps	1700 ps
Store	500 ps	300 ps	300 ps	500 ps		1600 ps
Branch	500 ps	300 ps	300 ps			1100 ps
Jump	500 ps	300 ps				800 ps

Clock cycle for the single-cycle processor = longest delay = 1700 ps = 1.7 ns

b) Compute the clock cycle and the average CPI for the **multi-cycle processor**. **(3 points)**

Clock cycle for the multi-cycle processor = max (500, 300, 100) = 500 ps

Average CPI = $(0.4 \times 4) + (0.2 \times 5) + (0.1 \times 4) + (0.2 \times 3) + (0.1 \times 2) = 3.8$

c) Determine quantitatively if there is a speedup when using the **multi-cycle processor**. **(2 points)**

Speedup of multi-cycle over single-cycle = $(1700 \times 1) / (500 \times 3.8) = 0.895$

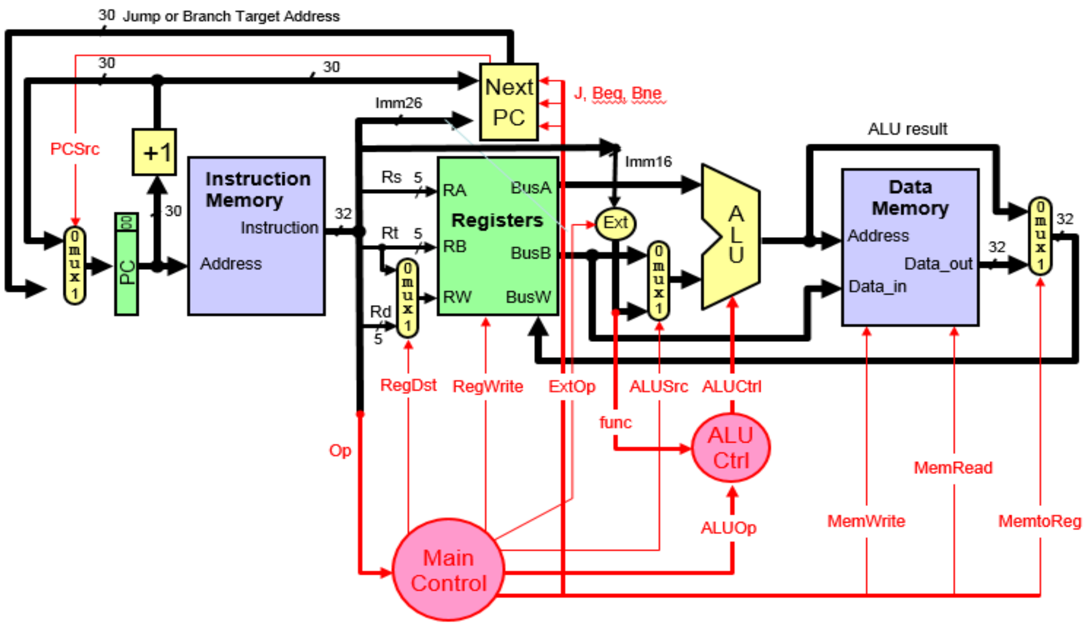
$\Rightarrow$ No speedup!

d) If we want to improve the execution time of a program on the **single-cycle processor** by a factor of 2, determine quantitatively whether enhancing the ALU unit can or cannot help achieve an overall speedup of 2 for the program execution time. **(2 points)**

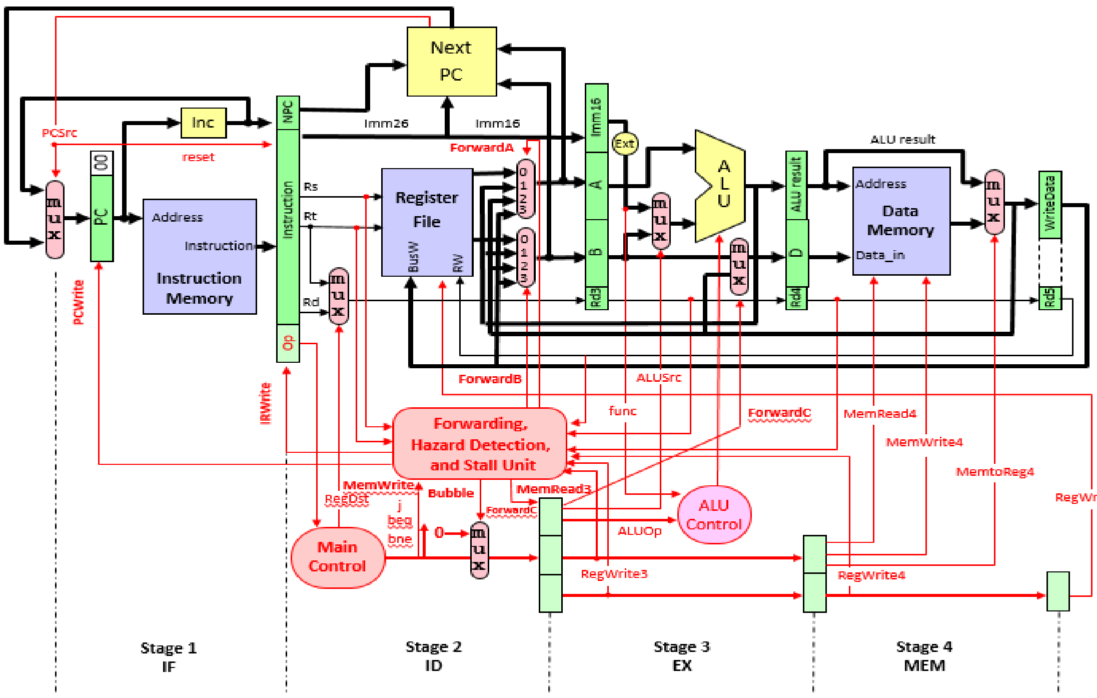
Using Amdahl's law: $2 = 1/(0.4/s + 0.6) \Rightarrow s = 0.4/(0.5 - 0.6) = -4 \Rightarrow$ Impossible!

[20 Points]

(Q2) Consider the single-cycle CPU design given below:



(i) Show the necessary modifications in the data path and control unit to implement this CPU as a 5-stage Pipeline (IF, ID, EX, MEM, WB) without consideration of data and control hazards. You can make changes in the given diagram but clarify the changes made and label them clearly. Assume that the NextPC block is independent of the ALU and that it has its own comparator and will be placed in the 2nd stage (i.e., ID stage) to reduce branch penalty. **(7 Points)**



(ii) Add the necessary changes to perform forwarding due to data hazards. Draw only the modified parts. Show the conditions that will be used for generating the **ForwardA** signals. **(6 Points)**

The conditions that will be used for generating the ForwardA signals:

```
If        ((Rs != 0) and (Rs == Rd3) and (RegWrite3))        ForwardA ← 1
Else if   ((Rs != 0) and (Rs == Rd4) and (RegWrite4))        ForwardA ← 2
Else if   ((Rs != 0) and (Rs == Rd5) and (RegWrite5))        ForwardA ← 3
Else                                                         ForwardA ← 0
```

(iii) Add the necessary changes to stall the pipeline due to data hazards. Draw only the modified parts. State the conditions for stalling the pipeline due to Data Hazards. (**4 Points)**

Condition for Stalling the pipeline due to Load Instruction:

```
if        ((MemRead3 == 1)     // Detect Load in EX stage
and (ForwardA==1 or ForwardB==1))          Stall  // RAW Hazard
```

OR:

```
if        ((MemRead3 == 1)
and (Rd3 ≠ 0) and ((Rs == Rd3) or (Rt == Rd3))) Stall
```

Stall means that the signals PCWrite=0 and IRWrite=0, which will freeze the content of PC and IR registers and bubble=1 which will introduce a bubble in stage 2 control register by setting the control signals to 0.

(iv) Add the necessary changes to stall the pipeline due to control hazards. Draw only the modified parts. State the conditions for stalling the pipeline due to Control Hazards. (**3 Points)**

Condition for Stalling the pipeline due to jump or taken branch Instruction:

Also, when PCSrc=1, reset=1 and the content of IR register will be reset to 0 to make the fetched instruction a NOP.
PCSrc=1 same as [(beq and Z) or (bne and Z') or J]

[15 Points]

(Q3)

(i) Consider the following MIPS assembly language code: **(7 Points)**

```
I1:    ADDI $s0, $0, 10
I2:    LW   $s1, 0($s0)
I3:    ADD  $s0, $s0, $s1
I4:    SLL  $s1, $s0, 4
I5:    LW   $s1, 4($s0)
I6:    ADDI $s1, $s1, -1
I7:    SW   $s1, 4($s0)
```

Complete the following table showing the timing of the above code on the 5-stage pipeline MIPS processor (IF, ID, EX, MEM, WB) assuming that it supports **forwarding** and **pipeline stall**. Draw an arrow showing forwarding between the stage that provides the data and the stage that receives the data. Show all stall cycles by placing an X in the box to represent a stall cycle. Determine the number of clock cycles to execute this code.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
I1: ADDI	IF	ID	EX	-	WB										
I2: LW		IF	ID	EX	M	WB									
I3: ADD			IF	X	ID	EX	-	WB							
I4: SLL					IF	ID	EX	-	WB						
I5: LW						IF	ID	EX	M	WB					
I6: ADDI							IF	X	ID	EX	-	WB			
I7: SW									IF	ID	EX	M	-		

The number of clock cycles to execute this code is 12.

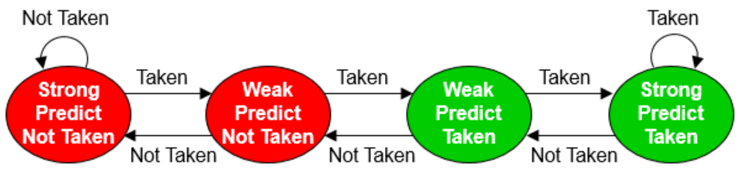
(ii) A sequence of three branches are shown in the first column of the table below with their respective PC value, branch target address, and next PC. These branches are executed in four passes (pass 1 to 4). The actual branch outcomes of each branch in each pass are shown where T represents a branch taken and NT represents a branch not taken. Assume a Branch Target Buffer (BTB) is used for early prediction of taken branches.

Four Branches with their respective PC values, branch target address, and next PC.	Branch outcomes in four passes			
	Actual Branch outcome In pass 1	Actual Branch outcome In pass 2	Actual Branch outcome In pass 3	Actual Branch outcome In pass 4
00010 Beq -,-, (target=00100) 00011 …. ..	T	T	T	NT
00101 Beq -,-, (target=00111) 00110 …. ..	NT	T	NT	T
01011 Beq -,-, (target=01101) 01100 …. ..	T	NT	T	T

a) Fill in the BTB entries for PC, target and initial prediction (T for all) for the three branches above. Fill in the prediction in BTB table **after** each pass (1 to 4) by assuming a 1-bit prediction. Then, compute the probability of correct prediction. **(3 points)**

BTB			BTB Prediction just **after** pass k (k = 1 to 4)			
PC	Target	Prediction initial	Prediction pass 1	Prediction pass 2	Prediction pass 3	Prediction pass 4
00010	00100	T	T	T	T	NT
00101	00111	T	NT	T	NT	T
01000	10010	T	T	NT	T	T
Probability of Correct Prediction = (3+0+2)/12 = 0.417						

b) Repeat the question above by predicting the branch outcome using a 2-bit saturating counter (given below). Denote by NT1 and T1 the weak NT and weak T, respectively. Assume that the predictor is initialized to T1 (weak predict taken). Then, compute the probability of correct prediction. **(3 points)**



BTB			BTB Prediction just **after** pass k (1 to 4)			
PC	Target	Prediction initial	Prediction pass 1	Prediction pass 2	Prediction pass 3	Prediction pass 4
00010	00100	T1	T	T	T	T1
00101	00111	T1	NT1	T1	NT1	T1
01000	10010	T1	T	T1	T	T
Probability of Correct Prediction = (3+0+3)/12 = 0.5						

c) Assume that a correctly predicted branch incurs zero stalls and a mis-predicted branch incurs 2 stalls. Evaluate the average number of stalls per instruction for using the 1-bit and 2-bit predictors if 15% of the instructions are branches. **(2 points)**

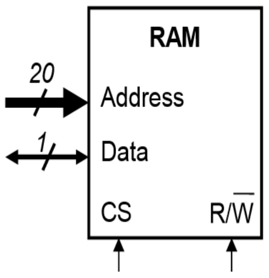
<u>For 1-bit prediction</u>:
Average number of stalls per instruction = $0.15 \times (1 - 0.417) \times 2 = 0.1749$

<u>For 2-bit prediction</u>:
Average number of stalls per instruction = $0.15 \times (1 - 0.5) \times 2 = 0.15$

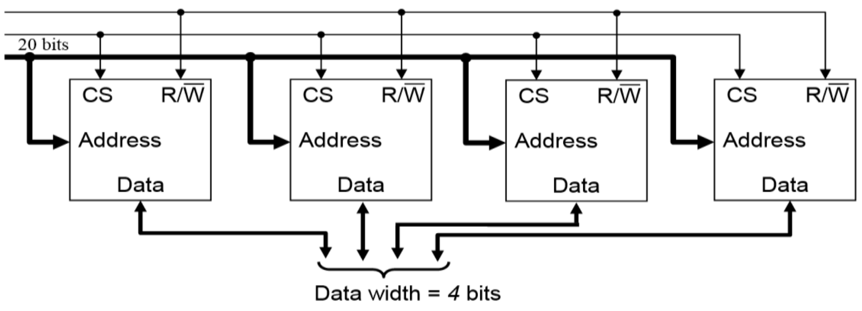
[22 Points]

(Q4)

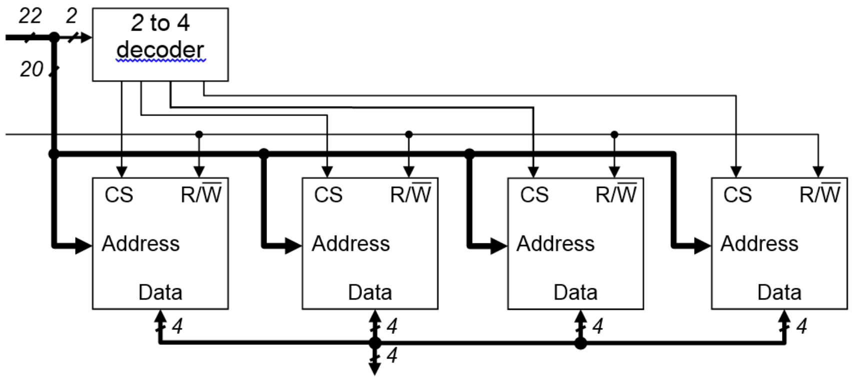
(i) Given a 1M x 1 memory block as shown below. Use this block to implement a 4M x 4 memory block. **(4 Points)**



First, we design a block of RAM of size 1M x 4 as follows:



Then, we design a block of RAM of size 4M x 4 using the above block as follows:



(ii) Assume that you have a 32-bit address and a cache with **8K byte data size** (not including tag and valid bits).

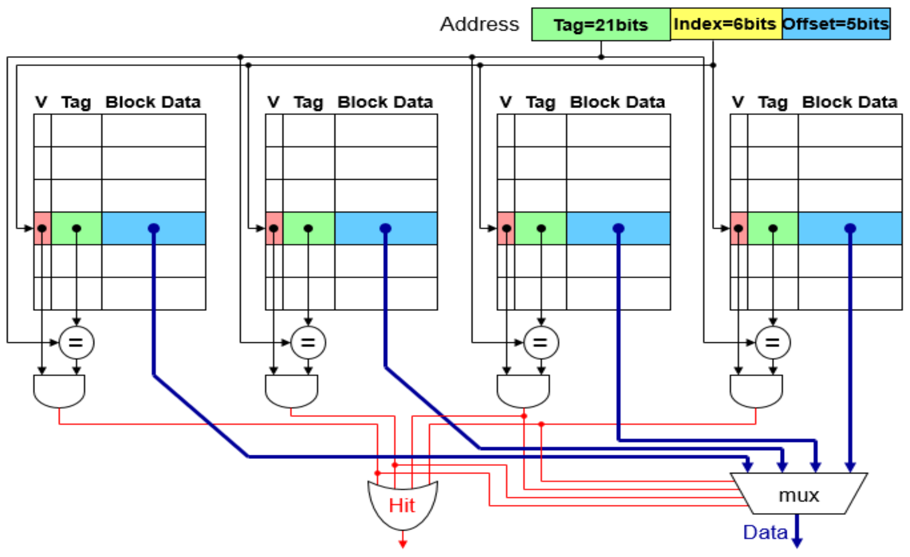
a) Assuming that the cache is organized as **direct-mapped** with a **32-byte block size**, determine the number of bits in the offset, index and tag fields. **(3 Points)**

Offset = $\log_2$ (block size) = $\log_2 32$ = 5 bits
Index = $\log_2$ (# locations) = $\log_2$ (8K/32) = 8 bits
Tag = 32 – (5+8) = 19 bits

b) Assuming that the cache is organized as **four-way set associative** with a **32-byte block size**, determine the number of bits in the offset, index and tag fields. **(2 Points)**

Offset = $\log_2$ (block size) = $\log_2 32$ = 5 bits
Index = $\log_2$ (# locations) = $\log_2 (8K/(32*4))$ = 6 bits
Tag = 32 – (5+6) = 21 bits

c) Show the organization of a **cache organized as four-way set associative** with a **32-byte block size**. **(4 Points)**



(iii) A processor runs at 2.0 GHz and has a CPI=1.9 for a perfect cache (i.e. without including the stall cycles due to cache misses). Assume that load and store instructions are 20% of the instructions. The processor has an I-cache with a 3% miss rate and a D-cache with 6% miss rate. The hit time is 1 clock cycle for both caches. Assume that the time required to transfer a block of data from the RAM to the cache, i.e. miss penalty, is 25 ns.

a) What is the number of stall cycles per instruction and the overall CPI? **(3 Points)**

Miss penalty in clock cycles = $25 * 10^{-9} * 2.0 * 10^{9} = 50$ cycles
Number of stall cycles per instruction = 0.03*50+0.20*0.06*50 = 2.1
Overall CPI = 1.9 + 2.1 = 4.0

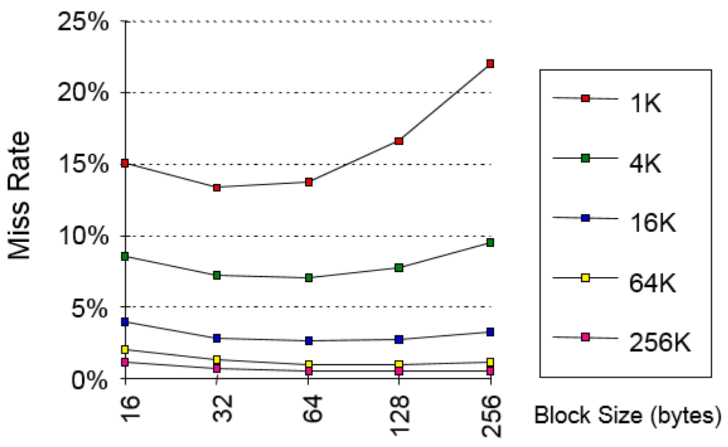
b) What is the average memory access time (AMAT) in ns? **(4 Points)**

AMAT(IC) = Hit time(IC) + Miss rate(IC) × Miss penalty
= 0.5 ns + 0.03 * 25ns = 1.25 ns

AMAT(DC) = Hit time(DC) + Miss rate(DC) × Miss penalty
= 0.5 ns + 0.06 * 25ns = 2.0 ns

AMAT = $1/(1+P_{LS})$ * AMAT(IC) + $P_{LS}/(1+P_{LS})$ * AMAT(DC)
= 1/(1+0.2) * 1.25 ns + 0.20/(1+0.20)*2.0 ns
= 1.042 ns + 0.333 = 1.375 ns

c) Consider the given figure below which plots the Miss Rate vs. the Block Size for various cache sizes. Explain why increasing the block size reduces the miss rate initially but after a certain point the miss rate increases by increasing the block size. **(2 Points)**



The miss rate decreases initially as the decrease in the number of compulsory misses is higher than the increase in the number of conflict misses. However, after increasing the block size beyond a certain point the miss rate increases when the increase in the number of conflict misses is larger than the decrease in the number of compulsory misses.