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

ICS 233: COMPUTER ARCHITECTURE & ASSEMBLY LANGUAGE

Term 142 (Spring 2014-2015)
Final Exam
Thursday May 21, 2015

Time: 150 minutes, Total Pages: 12

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	16	
Q2	24	
Q3	12	
Q4	15	
Q5	16	
Total	83	

Dr. Aiman El-Maleh
Dr. Mayez Al-Muhammad

[16 Points]

(Q1)

(i) You are required to enhance a computer performance for running a given program, and there are two possible improvements that can be made: improve execution time of multiply instructions and improve memory access time. Your program takes 120 seconds to execute. Of this time, 25% is used for multiplication instructions, 35% for memory access instructions, and 40% for other tasks. Suppose the execution time of multiplication instructions was improved by a factor of 2, what is the speedup factor required for memory access instructions to have an overall speedup by a factor of 1.5. What will the new execution time be? **(5 points)**

New execution time = Old Execution time /speedup = 120/1.5 = 80s.

Old execution time for multiply = 0.25*120 = 30 s
Old execution time for memory access = 0.35*120 = 42 s
Old execution time for other tasks = 48 s

New execution time = Old execution time for multiply/2 + old execution time for memory access/S + 48
80 = 30/2 + 42/S + 48
⇨ 42/S = 17 => S = 42/17 = 2.471

Thus, the speedup factor required for memory access instructions is 2.471 to have an overall speedup by a factor of 1.5

OR:

Speedup = 1 / (f1/s1 + f2/s2 + (1-f1-f2)
1.5 = 1/ (0.25/2 + 0.35/s2 + 0.4) => (0.25/2 + 0.35/s2 + 0.4) = 1/1.5=0.667
0.35/s2 = 0.14167 => s2 = 0.35/0.14167 = 2.471

(ii) Discuss how the compiler could affect the execution time of a high-level program. **(3 points)**

The compiler could affect the execution time by affecting the number of executed instructions in a program and also the average CPI as this depends on the instruction types executed.

(iii) Given the following instruction mix of a program on a RISC processor:

(8 points)

Class	CPI	Frequency
ALU	2	50%
Branch	2	20%
Jump	1	10%
Load	4	12%
Store	3	8%

(i) What is the average CPI?

Average CPI=2*0.5 + 2*0.2+1*0.10+4*0.12+3*0.08=2.22

(ii) What is the percent of time used by each instruction class?

Class	Percentage of Time
ALU	2*0.5/2.22=1/2.22=45.05%
Branch	2*0.2/2.22=0.4/2.22=18.02%
Jump	1*0.10/2.22=0.10/2.22=4.50%
Load	4*0.12/2.22=0.48/2.22=21.62%
Store	3*0.08/2.22=0.24/2.22=10.81%

(iii) How much faster would the program run if load time is reduced to 3 cycles, and two ALU instructions could be executed at once, assuming that the cycle time has increased by 4% and the instruction count has increased by 8%?

New Average CPI=**1***0.5 + 2*0.2+1*0.10+**3***0.12+3*0.08=1.6

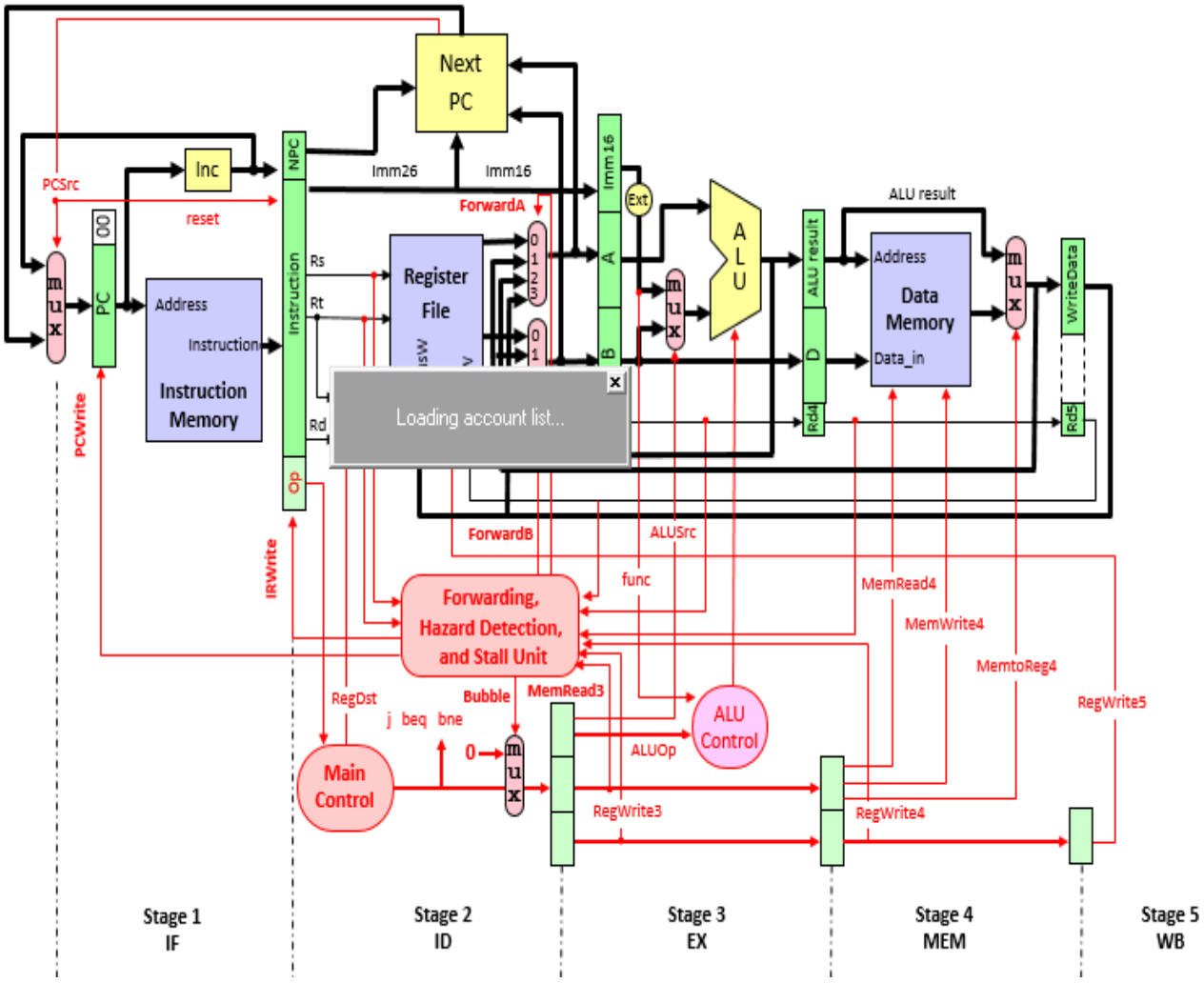
Speedup = Old Execution Time / New Execution Time
= IC * 2.22 * Cycle Time / 1.08* IC * 1.6 * 1.04 * Cycle Time

=2.22/1.08*1.6*1.04=2.22/1.797=1.235

[24 Points]

(Q2)

(i) Consider the 5-stage pipelined CPU design given below.



a) Show the conditions that will be used for generating the ForwardA signals.

b) Show the control signals that will be used for stalling the pipeline due to data and control hazards along with their conditions.

c) Add the necessary changes to the design, on the given diagram, to allow it to handle data hazards due to load instructions and control hazards.

(12 Points)

i) 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
```

ii) **Condition for Stalling the pipeline:**

1. 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.

2. Stalling the Pipeline due to 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]

(ii) Consider the following MIPS assembly language code: **(12 Points)**

```
I1:     ADDI $s0, $0, 10
I2:     ADDI $s1, $0, 5
I3:     SLL  $s0, $s0, 4
I4:     LW   $s2, 0($s0)
I5:     ADD  $s2, $s2, $s1
I6:     SW   $s2, 4($s0)
```

a) Determine all RAW data hazards in the above program.

RAW Hazard	Instruction 1	Instruction 2
RAW 1	I1	I3
RAW 2	I3	I4
RAW 3	I4	I5
RAW 4	I5	I6
RAW 5		
RAW 6		

b) Complete the following table showing the timing of the above code on the 5-stage pipeline given in part (i) (IF, ID, EX, MEM, WB) that supports **forwarding**. Draw an arrow showing forwarding between the stage that provides the data and the stage that receives the data. Show all stall cycles (draw 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: ADDI		**IF**	**ID**	**EX**	-	**WB**									
I3: SLL			**IF**	**ID**	**EX**	-	**WB**								
I4: LW				**IF**	**ID**	**EX**	**MEM**	**WB**							
I5: ADD					**IF**	**X**	**ID**	**EX**	-	**WB**					
I6: SW							**IF**	**ID**	**EX**	**MEM**	-				

The number of clock cycles needed to execute the program is 11 clock cycles.

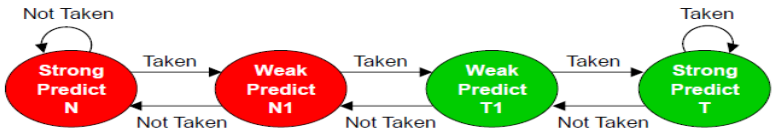
[12 Points]

(Q3) A series of three conditional branches B1 to B3 are sequentially executed one after the other. When a branch is encountered, it is executed six times. The outcome of each branch (called actual) is listed in the table below as taken (T) or not taken (N).

(i) Assume a 1-bit history predictor is used which is initialized to N. Fill in the prediction entry for each branch and evaluate the probability of correct prediction. **(4 Points)**

Conditional Branch	
B1: Prediction	N N T N N T
B1: Actual	N T N N T T
B2: Prediction	N T N T N T
B2: Actual	T N T N T N
B3: Prediction	N T T N N T
B3: Actual	T T N N T T
Probability of Correct Prediction	=6/18=33.3%

(ii) Assume the 2-bit history predictor, given below, is used. Denote by T1 the weakly predict Taken and N1 the weakly predict not taken. Assume that the predictor is initialized to N1 (weak predict not taken). Fill in the prediction entry for each branch and evaluate the probability of correct prediction. **(Note: fill your predictions as T, T1, N, N1) (4 Points)**



Conditional Branch	
B1: Prediction	N1 N N1 N N N1
B1: Actual	N T N N T T
B2: Prediction	N1 T1 N1 T1 N1 T1
B2: Actual	T N T N T N
B3: Prediction	N1 T1 T T1 N1 T1
B3: Actual	T T N N T T
Probability of Correct Prediction	=5/18=27.78%

(iii) Assume that a correctly predicted branch incurs zero stalls and a mis-predicted branch incurs 3 stalls. Evaluate the average number of stalls per instruction for using the 1-bit and 2-bit predictors if 20% of the instructions are branches.

(4 Points)

For 1-bit prediction:
Average number of stalls per instruction = 0.2 * 0.667 * 3 = 0.4002

For 2-bit prediction:
Average number of stalls per instruction = 0.2 * 0.722 * 3 = 0.4332

[15 Points]

(Q4) A CPU has an I-Cache, a D-cache and a main-memory. For some reference program, the instruction distribution is as follows: (1) R-type instructions are 40%, (2) Load instructions are 25%, (3) Store instructions are 10%, and (4) and the rest are for the branch instructions. The CPU clock frequency is 2 GHz. The memory specifications are:

1. I-Cache: the cache hit time is 1 ns and the probability of a hit is 0.96
2. D-Cache: the cache hit time is 2 ns and the probability of a hit is 0.92
3. Main memory: the access time is 45 ns.
4. Assume that CPI= 2 clocks when all memory references are hits.

(i) Evaluate the main memory access time in clock cycles. **(1 Point)**

The main memory access time in clock cycles = $45 * 10^{-9} * 2 * 10^{9} = 90$ clock cycles

(ii) Evaluate the average number of stalls per instruction and the CPI. **(4 Points)** **(<u>Note</u>: Do not mix adding time in ns with time in clock cycles.)**

Number of stall cycles per instruction = 0.04*90+0.35*0.08*90 = 6.12
Overall CPI = 2 + 6.12 = 8.12

(iii) Evaluate the average memory access time in ns. **(4 Points)**

AMAT(IC) = Hit time(IC) + Miss rate(IC) × Miss penalty
= 1 ns + 0.04 * 45ns = 2.8 ns

AMAT(DC) = Hit time(DC) + Miss rate(DC) × Miss penalty
= 2 ns + 0.08 * 45ns = 5.6 ns

$AMAT = 1/(1+P_{LS}) * AMAT(IC) + P_{LS}/(1+P_{LS}) * AMAT(DC)$
= 1 (1+0.35) * 2.8 ns + 0.35/(1+0.35)*5.6 ns
= 2.074 ns + 1.452 = 3.526 ns

(iv) To improve the access time of this memory system, the designer considered 3 options:

a) Use a larger I-cache for which the probability of a miss dropped by 50% and all other parameters remain the same.

Number of stall cycles per instruction = 0.02*90+0.35*0.08*90 = 4.32
Overall CPI = 2 + 4.32 = 6.32

b) Use an enhanced D-Cache for which the hit probability is increased by 4% and all other parameters remain the same.

Number of stall cycles per instruction = 0.04*90+0.35*0.0432*90 = 4.9608
Overall CPI = 2 + 4.9608= 6.9608

c) Use a faster Main memory for which the access time drops by 10% and all other parameters remain the same.

New access time = 45 ns *0.9 = 40.5 ns = 40.5 * 2 = 81 cycles
Number of stall cycles per instruction = 0.04*81+0.35*0.08*81 = 5.508
Overall CPI = 2 + 5.508= 7.508

Evaluate the CPI for each of above cases and determine the best option.

(6 Points)

Thus, the first option is the best as it has lower CPI.

[16 Points]

(Q5) A processor has a hierarchical memory system that consists of a 4-ways set-associative cache C and a main-memory M. The processor generates a 32-bit memory address. The cache has a total of 128 sets. The block size is 32 bytes, where each word is four bytes. M is byte organized, i.e. the processor address is relative to a byte but four consecutive bytes are read when loading a word. Answer each of the following questions:

(i) Find the number of bits in "Offset", "Index", and "TAG". **(3 Points)**

Offset = $\log_2$ (block size) = $\log_2$ 32 = 5 bits
Index = $\log_2$ (# sets) = $\log_2$ (128) = 7 bits
Tag= 32- (7+5) = 20 bits

(ii) Determine the cache capacity (excluding TAGs and controls) and main memory capacity in bytes. **(3 Points)**

Cache capacity = 128 * 4 * 32 = 2^{14}= 16 KByte

Main memory capacity = 2^{32}= 4 GByte

(iii) Shortly describe how to search for a word with a given 32-bit address in the above memory system by considering the case of a cache hit and a cache miss. You may describe the above using an algorithm, a block-diagram, or flow-chart. **(4 Points)**

1. From the address of the word, the index bits (7 bits) are used to index the cache.
2. The tag bits (20 bits) are compared with the 4 tag bits of the 4 blocks in the indexed entry in parallel.
3. If the tag bits match with the tag bits of one of the four blocks and the valid bit is set to 1, then there is a HIT and the word pointed block (offset) is transferred from the cache to the CPU.
4. If there is a miss then the block is read from main memory and stored in the cache entry if there is an empty block.
5. If there is no empty block, then one of the blocks has to be replaced depending on the used replacement method. The word pointed by offset in the block is sent to CPU.

(iv) Suppose we use a 2-way set associative cache instead of the above but with the same number of entries. **(2 Points)**
Is the cache hit time likely to increase or decrease: decrease
Is the hit probability likely to increase or decrease: decrease

(v) Suppose we use a fully associative cache instead of the above but with the same capacity. **(2 Points)**
Is the cache hit time likely to increase or decrease: increase
Is the hit probability likely to increase or decrease: increase

(vi) Suppose you observe that cache misses significantly increase beyond some data size. Select one of the following options to improve memory access time: **(2 Points)**

(a) Redesign the above cache with same capacity but larger degree of associativity.
(b) Select a new cache with same organization but with larger capacity.
(c) Redesign the above cache with same capacity but increasing the block size.