ECE 341 Final Exam Solution

Name: _____

Time allowed: 110 minutes
Total Points: 100
Points Scored: _____

Problem No. 1 (10 points)

For each of the following statements, indicate whether the statement is **TRUE** or **FALSE**. Each correct answer carries 2 points. The answer for the last statement counts towards extra credit.

(a) It is undesirable to use latches for building circuits that involve counters and shift registers. **TRUE**
(b) Using operand forwarding in the 5-stage pipelined processor eliminates all the pipeline stalls caused by data hazards. **FALSE**
(c) A write-back cache typically requires more writes to the main memory than a write-through cache. **FALSE**
(d) A Translation Lookaside Buffer (TLB) acts as a cache for the page table. **TRUE**
(e) Booth algorithm is most efficient when the multiplier has an alternating sequence of 1s and 0s. **FALSE**
(f) **(Extra credit question)** When using port-mapped I/O, I/O devices typically need fewer address lines as compared to using memory-mapped I/O. **TRUE**

Problem No. 2 (12 points)

Multiple possible answers are provided for each of the following questions. Only one answer is correct in each case. Mark the correct answer for each question. Each correct answer carries 4 points.

(a) Consider a 16M x 128 memory built by using 512K x 16 memory chips. How many rows of memory chips are needed?
   i. 8
   ii. 32
   iii. 64
   iv. 256

(b) A 16-bit **blocked carry-lookahead adder (CLA)** composed of four 4-bit CLA blocks is used to add two numbers X ($x_{15}x_{14}x_{13}.....x_1x_0$) and Y ($y_{15}y_{14}y_{13}....y_1y_0$). Under which of the following conditions is the carry-out c_{12} equal to 0?
   i. $c_0 = 0$, all the CLA blocks generate a carry, but none of the CLA blocks propagate a carry
   ii. $c_0 = 0$, all the CLA blocks propagate a carry, but none of the blocks generates a carry
   iii. $c_0 = 1$, all the CLA blocks propagate a carry, but none of the CLA blocks generates a carry

(c) A processor uses 46-bit virtual addresses with 2 MB pages. Which bits in the virtual address correspond to the "offset" field?

i. The most significant 34 bits
ii. The most significant 25 bits
iii. The least significant 12 bits
iv. The least significant 21 bits

(d) **(Extra credit question)** A **non-pipelined** 5-stage RISC processor running at 3 GHz is used to execute a program P_1. 40% of the instructions in P_1 are Load or Store instructions. Assume that the processor does not use a cache. All the instruction fetch and data read/write requests are served by main memory with a fixed latency of 6 clock cycles. How many instructions are completed by the processor in 10 seconds?

i. 2.5 billion
ii. 1.5 billion
iii. 100 million

Problem No. 3 (18 points)

Consider the following sequence of instructions being processed on the **pipelined** 5-stage RISC processor discussed in class:

Add R4, R2, R3
Store R5, #100(R4)
Load R6, #200(R4)
Subtract R7, R5, R6

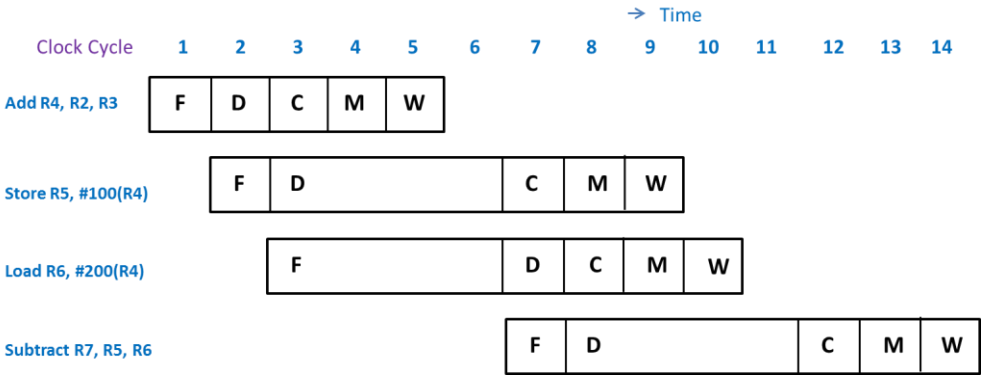
(a) **(6 points)** Identify all the data dependencies in the above instruction sequence. For each dependency, indicate the two instructions and the register that causes the dependency.

Solution:

There are three data dependencies in this instruction sequence:
(i) Store instruction depends on Add instruction for register R4
(ii) Load instruction depends on Add instruction for register R4
(iii) Subtract instruction depends on Load instruction for register R6

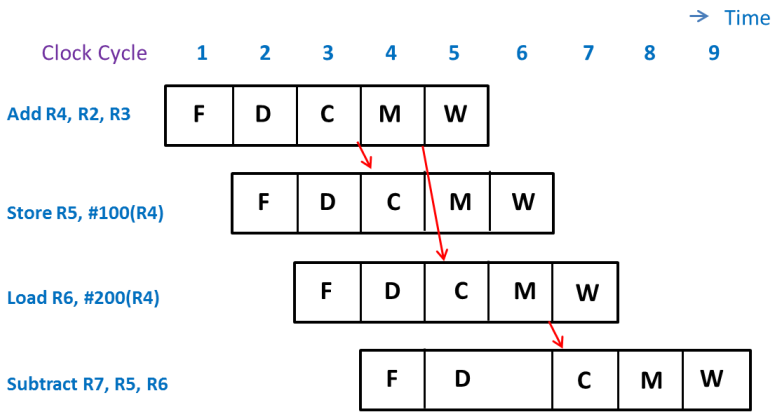
(b) **(6 points)** Assume that the pipeline does not use operand forwarding. Also assume that the only sources of pipeline stalls are the data hazards. Draw a diagram that represents instruction flow through the pipeline during each clock cycle. How long does it take for the instruction sequence to complete?

Solution:



(c) **(6 points)** Now, assume that the pipeline uses operand forwarding. There are separate forwarding paths from the outputs of stage-3 and stage-4 to the input of stage-3. Draw a diagram that represents the flow of instructions through the pipeline during each clock cycle. Indicate operand forwarding by arrows.

Solution:

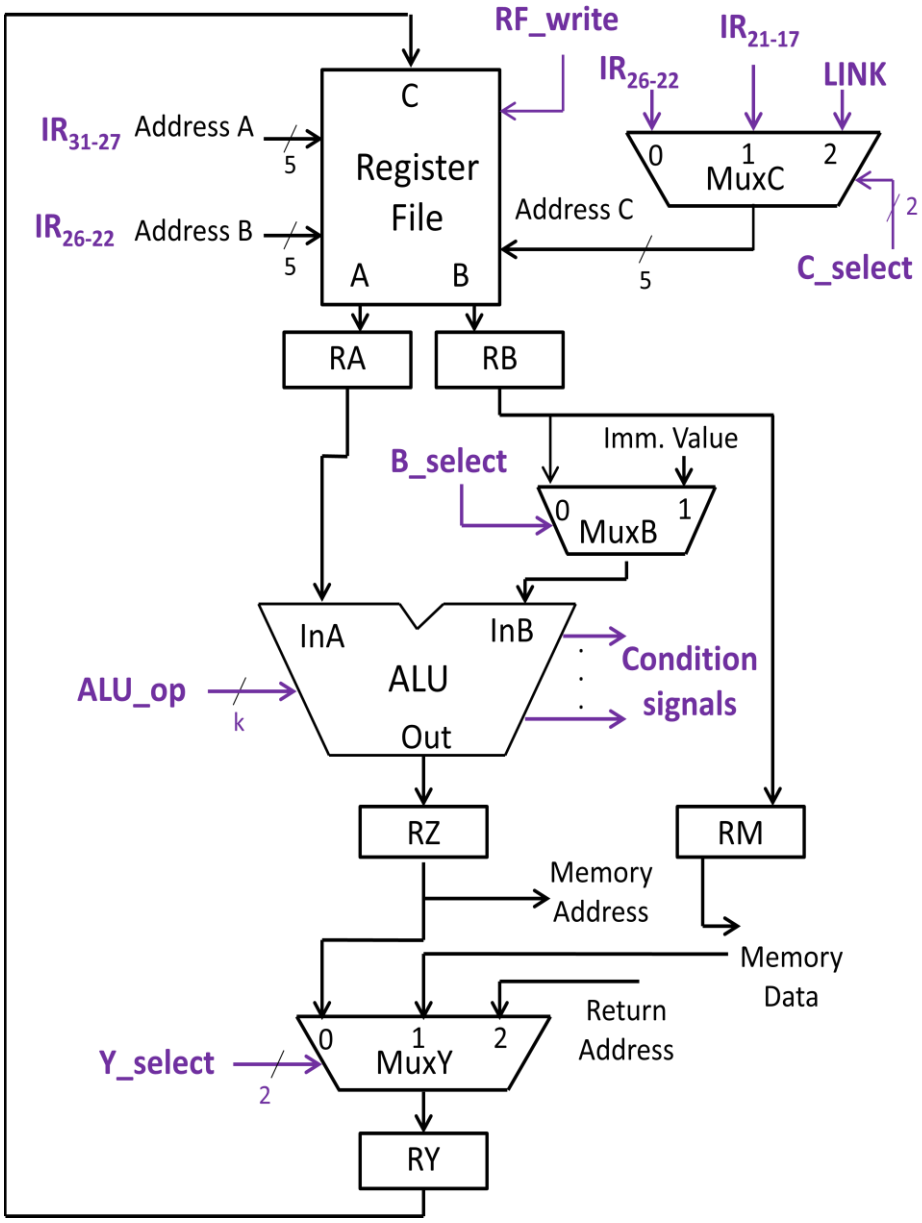


Problem No. 4 (12 points)

(a) **(6 points)** The 5-stage RISC processor discussed in class is used to execute the following sequence of instructions, one after the other:

Instruction 1: Add R4, R2, R3
Instruction 2: Load R5, #100(R4)
Instruction 3: Call_Register R7

The processor data path with all the control signals is shown in the following figure:



Write down the values of the following control signals for each of the three instructions:

I. ***Y_select*** during stage-4 of instruction processing
II. ***RF_write*** during stage-5 of instruction processing
III. ***C_select*** during stage-5 of instruction processing

Solution:

i. *Y_select* has values of **00**, **01** and **10** for instructions 1, 2, and 3, respectively.
ii. *RF_write* has a value of **1** for each of the three instructions.
iii. *C_select* has values of **01**, **00** and **10** for instructions 1, 2, and 3, respectively.

(b) **(6 points)** Assume that the initial contents of registers R2, R3 and R4 are 200, 400 and 500, respectively. Also assume that the initial contents of memory addresses 500, 600 and 700 are 10, 20 and 30, respectively. Write down the contents of inter-stage register ***RZ*** after the completion of stage-3 for instructions 1 and 2. Also write down the contents of register **R5** after the completion of above instruction sequence.

Solution:

For instruction 1:
Contents of *RZ* = [R2] + [R3] = 200 + 400 = **600.** Contents of R4 become 600 after instruction 1.

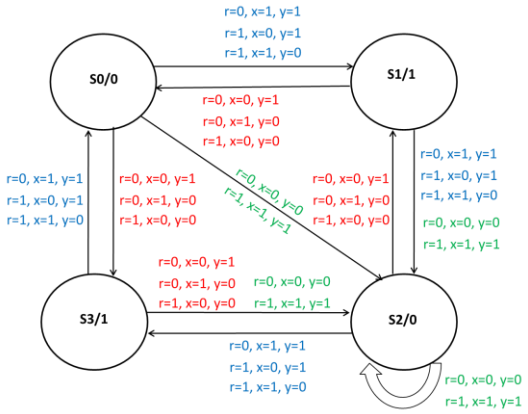
For instruction 2:
Contents of *RZ* = [R4] + 100 = 600 + 100 = **700**

Contents of R5 = Contents of memory address [700] = **30**

Problem No. 5 (15 points)

(a) **(6 points)** You are required to design a 2-bit synchronous counter by using a finite state machine. The counter has three external input signals *r, x,* and *y*, which dictate the operation of the counter as follows: (i) If exactly one of the three inputs is a zero, the counter counts *up*, (ii) If exactly two of the three inputs are zeroes, the counter counts *down*, (iii) If all the three inputs have identical values, the counter is set to a count of "2", irrespective of its previous state. The counter also has an output signal *z*, which is equal to 1 only if the present value of the counter is an odd number. Draw the state diagram for this state machine.

Solution:



(b) **(4 points)** A 4-bit carry-looakhead adder (CLA) is used to add two numbers X = 1101 and Y = 0101 with an external carry-in c_0 = 1. Compute the value of c_4 by using the carry-lookahead equations.

Solution:

For a 4-bit CLA, c4 is given by:

$c_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0c_0$

$G_0 = x_0$ AND y_0 = 1 AND 1 = 1

$G_1 = x_1$ AND y_1 = 0 AND 0 = 0

$G_2 = x_2$ AND y_2 = 1 AND 1 = 1

$G_3 = x_3$ AND y_3 = 1 AND 0 = 0

$P_0 = x_0$ OR y_0 = 1 OR 1 = 1

$P_1 = x_1$ OR y_1 = 0 OR 0 = 0

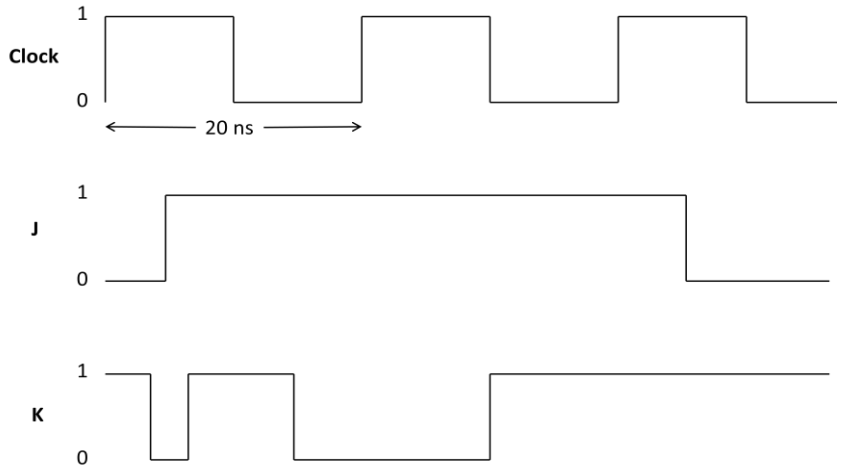
$P_2 = x_2$ OR y_2 = 1 OR 1 = 1

$P_3 = x_3$ OR y_3 = 1 OR 0 = 1

Therefore:

$c_4 = 0 + 1.1 + 1.1.0 + 1.1.0.1 + 1.1.0.1.1 = 0 + 1 + 0 + 0 + 0 =$ **1**

(c) **(5 points)** The input waveforms for a positive edge-triggered JK flip flop are shown in the following figure. Assume that each waveform starts at time = 0 and output Q of the flip-flop has a logic value of "0" at time = 0. What is the logic value of output Q at the following points of time: (i) 18 ns, (ii) 35 ns, (iii) 50 ns?



Solution:

i. At t = 18 ns, **Q = 0** (Q retains its initial value until there is a positive clock edge)

ii. At t = 35 ns, **Q = 1** (Positive clock edge arrives at t = 20 ns and J= 1, K = 0)

iii. At t = 50 ns, **Q = 0** (Q toggles its value because there is a positive clock edge at 40 ns and J = 1, K = 1)

Problem No. 6 (15 points)

A 5-stage pipelined RISC processor C_1 running at 3 GHz is used to execute a program P_1. The instruction statistics for P_1 are as follows:

Branches: 20%
Loads: 20%
Stores: 10%
Arithmetic Instructions: 50%

Assume that the program P_1 has no data dependencies. C_1 uses a dynamic branch predictor and a branch target buffer to predict the branch instructions. The computation of actual branch outcomes is carried out in the "compute" stage (stage-3). Also assume that C_1 uses a cache such that **100%** of the instruction fetches and data writes hit in the cache, whereas 95 **%** of the data reads hit in the cache. The penalty to access the main memory for a cache miss is 15 cycles.

To execute the program completely, the processor C_1 needs to run 12 billion instructions. A customer requires that the program must be completed in less than 5 seconds. What is the minimum branch prediction accuracy which would satisfy this requirement?

Solution:

Clock Rate "R" = 3 GHz = $3*10^9$ Hz
We need to consider stall cycles due to both the cache misses and branch mispredictions.

For cache misses:
Since 100% of the instruction fetches and data writes hit in the cache, the only source of cache misses are the data reads caused by the Load instructions.
Percentage of Load instructions = 20%
Percentage of data reads that hit in the cache = 95%
Therefore, percentage of data accesses that miss in the cache = (100 - 95) % = 5%
Cache miss penalty = 15 cycles
Therefore, δ_{miss} = stall frequency * stall penalty = 0.2 * 0.05 * 15 = 0.15

For branch mispredictions:
Percentage of branch instructions = 20%
Since branch computation is carried out in "Compute" stage (stage-3), stall penalty = 2 cycles
Let us assume that, branch prediction accuracy = x%
Percentage of branches that are predicted inaccurately = (100-x) %
$\delta_{branch\_penalty}$ = stall frequency * stall penalty = 0.2 * ((100-x)/100) * 2 = 0.4 – 0.004x

Adding the two different sources of stall cycles:
Total stall cycles per instruction $\delta = \delta_{miss} + \delta_{branch\_penalty}$ = 0.15 + 0.4 – 0.004x = 0.55 – 0.004x

Throughput "P_p" = R / (1+δ) = $(3*10^9)/(1 + 0.55 - 0.004x) = (3*10^9)/(1.55 – 0.004x)$
Required throughput = 12 billion instructions / 5 seconds = $2.4 * 10^9$ instructions/second
To satisfy the performance requirement:
$(3*10^9)/(1.55 – 0.004x) >= 2.4*10^9$
Solving for "x" in the above equation yields x >= 75
Therefore, the required minimum branch prediction accuracy is **75 %**

Problem No. 7 (18 points)

(a) **(6 points)** A computer system uses 36-bit memory addresses. It has a 16M-byte 16-way set-associative cache, with 128 bytes per cache block. Assume that the size of each memory word is 1 byte. Calculate the number of bits in each of the *Tag*, *Set*, and *Word* fields of the memory address.

Solution:

Block size = 128 bytes = 2^7 bytes = 2^7 words
Therefore, **Number of bits in the *Word* field = 7**
Cache size = 16 M-byte = 2^{24} bytes
Number of cache blocks per set = 16 = 2^4
Number of sets = Cache size / (Block size * Number of blocks per set) = $2^{24}/(2^7 * 2^4) = 2^{13}$
Therefore, **Number of bits in the *Set* field = 13**
Total number of address bits = 36
Therefore, **Number of bits in the *Tag* field =** 36 - 7 - 13 **= 16**

(b) **(12 points)** A processor that uses 10-bit memory addresses has a small 2-way set-associative cache capable of holding four cache blocks. Each cache block consists of 16 words. The cache uses the least-recently-used (LRU) replacement algorithm. Assume that the initial tag values for each cache block are as follows:

Set	0		1	
Block	0	1	0	1
Tag	00001	Invalid	Invalid	00010

The processor reads data sequentially from the following decimal addresses: 36, 10, 80, 64, 36

For each of the above addresses, indicate whether the cache access will result in a hit or a miss.

Solution:

Block size = 16 words = 2^4 words => No. of bits in the *Word* field = 4
Number of sets = Number of blocks / Associativity = 4/2 = 2 (also shown in figure)
Therefore, Number of bits in the *Set* field = 1
Total number of address bits = 8
Therefore, Number of bits in the *Tag* field = 10 - 4 - 1 = 5
For a given 10-bit address, the 5 most significant bits represent the *Tag*, the next bit represents the *Set*, and the 4 least significant bits represent the *Word*.

Access # 1:
Address = $(36)_{10}$ = $(0000100100)_2$. For this address, *Tag* = 00001, *Set* = 0
Tag for block-0 in set 0 matches the address tag => cache **hit**

Access # 2:

Address = $(10)_{10}$ = $(0000001010)_2$. For this address, *Tag* = 00000, *Set* = 0

Neither of the tags in set 0 match the address tag => cache **miss**

Since block-1 in set 0 is empty (invalid), the new block is brought there.

Tag field for block-1 in set 0 is set to 00000

Set	0		1	
Block	0	1	0	1
Tag	00001	**00000**	Invalid	Invalid

Access # 3:

Address = $(80)_{10}$ = $(0001010000)_2$. For this address, *Tag* = 00010, *Set* = 1

Neither of the tags in set 1 match the address tag => cache **miss**

Both the blocks in set 1 are empty (invalid). Block-0 is subsequently filled.

Tag field for block-0 in set 1 is set to 00010

Set	0		1	
Block	0	1	0	1
Tag	00001	00000	**00010**	Invalid

Access # 4:

Address = $(64)_{10}$ = $(0001000000)_2$. For this address, *Tag* = 00010, *Set* = 0

Neither of the tags in set 0 match the incoming address tag => cache **miss**

The LRU replacement algorithm will replace block-0 (the LRU block). Therefore, the Tag field for block-0 in set 0 is set to 00010

Set	0		1	
Block	0	1	0	1
Tag	**00010**	00000	00010	Invalid

Access # 5:

Address = $(36)_{10}$ = $(0000100100)_2$. For this address, *Tag* = 00001, *Set* = 0

Neither of the tags in set 0 match the address tag => cache **miss**