ECE 341 Final Exam Solution

Name: _____

Time allowed: 2 hours
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) Any logic function expressed in a sum-of-products form can be implemented by using NAND gates. **TRUE**
(b) Pipelining reduces the time needed to process a single instruction. **FALSE**
(c) Increasing the cache block size while keeping the total cache size constant can sometimes hurt performance. **TRUE**
(d) The cost per bit for DRAM is lower than the cost per bit for SRAM. **TRUE**
(e) When interrupts are used to synchronize data transfer between a processor and an I/O device, the processor needs to repeatedly monitor the status of I/O device. **FALSE**
(f) **(Extra credit question)** In port-mapped I/O, any machine instruction that can access memory can also be used to transfer data to an I/O device. **FALSE**

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. The answer for the last question counts towards extra credit.

(a) Consider a 8M x 64 memory built by using 512K x 16 memory chips. What is the total number of memory chips needed?
   i. 16
   ii. 32
   iii. 64
   iv. 128

(b) A processor uses 44-bit virtual addresses with 4 kB pages. Which bits in the virtual address correspond to the "virtual page number" field?
   i. The least significant 12 bits
   ii. The most significant 12 bits
   iii. The least significant 32 bits
   iv. The most significant 32 bits

(c) You are required to implement a logic function $f = xy + yz$ in its **minimal** form. You can use any combination of 2-input logic gates. What is the **minimum** number of logic gates required to implement f?

i. 1 gate
ii. **2 gates**
iii. 4 gates

(d) **(Extra credit question)** A **non-pipelined** 5-stage RISC processor running at 2.5 GHz is used to execute a program P_1 with 1 billion instructions. 50% 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 long does it take for the processor to complete the program P_1?

i. 1 second
ii. 2 seconds
iii. **5 seconds**

Problem No. 3 (18 points)

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

```
Load R4, #100(R2)
Add R5, R2, R3
Subtract R6, R4, R5
And R7, R2, R5
```

(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:

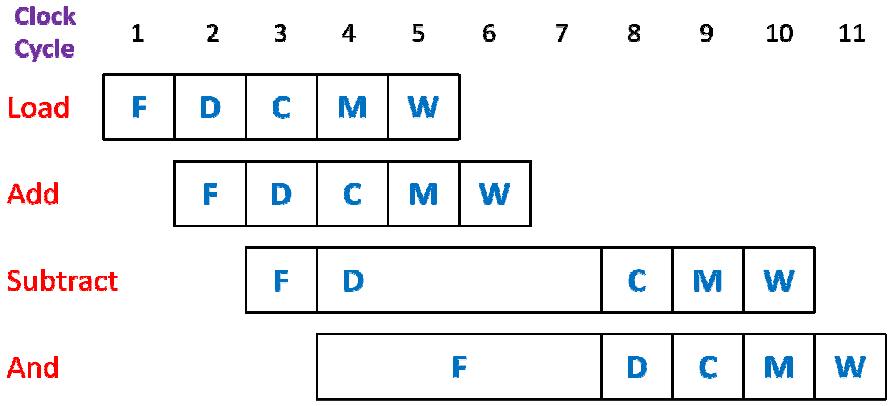
There are three data dependencies in this instruction sequence:

(i) Subtract instruction depends on Load instruction for register R4
(ii) Subtract instruction depends on Add instruction for register R5
(iii) And instruction depends on Add instruction for register R5

(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:

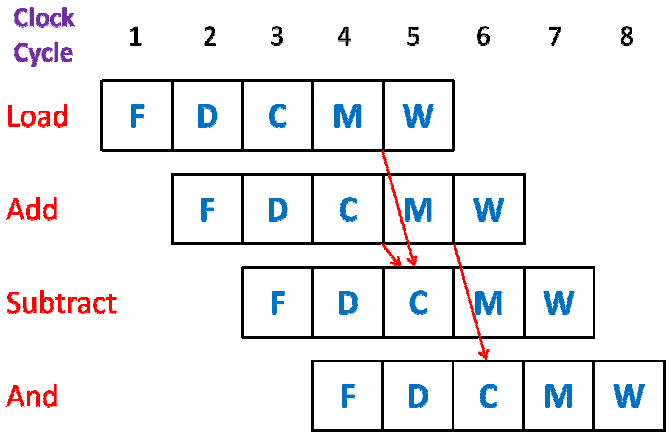
The following diagram shows the instruction flow through the pipeline. As shown in the diagram, the instruction sequence is completed in 11 cycles.



(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:

The following diagram shows the instruction flow through the pipeline in the presence of operand forwarding:

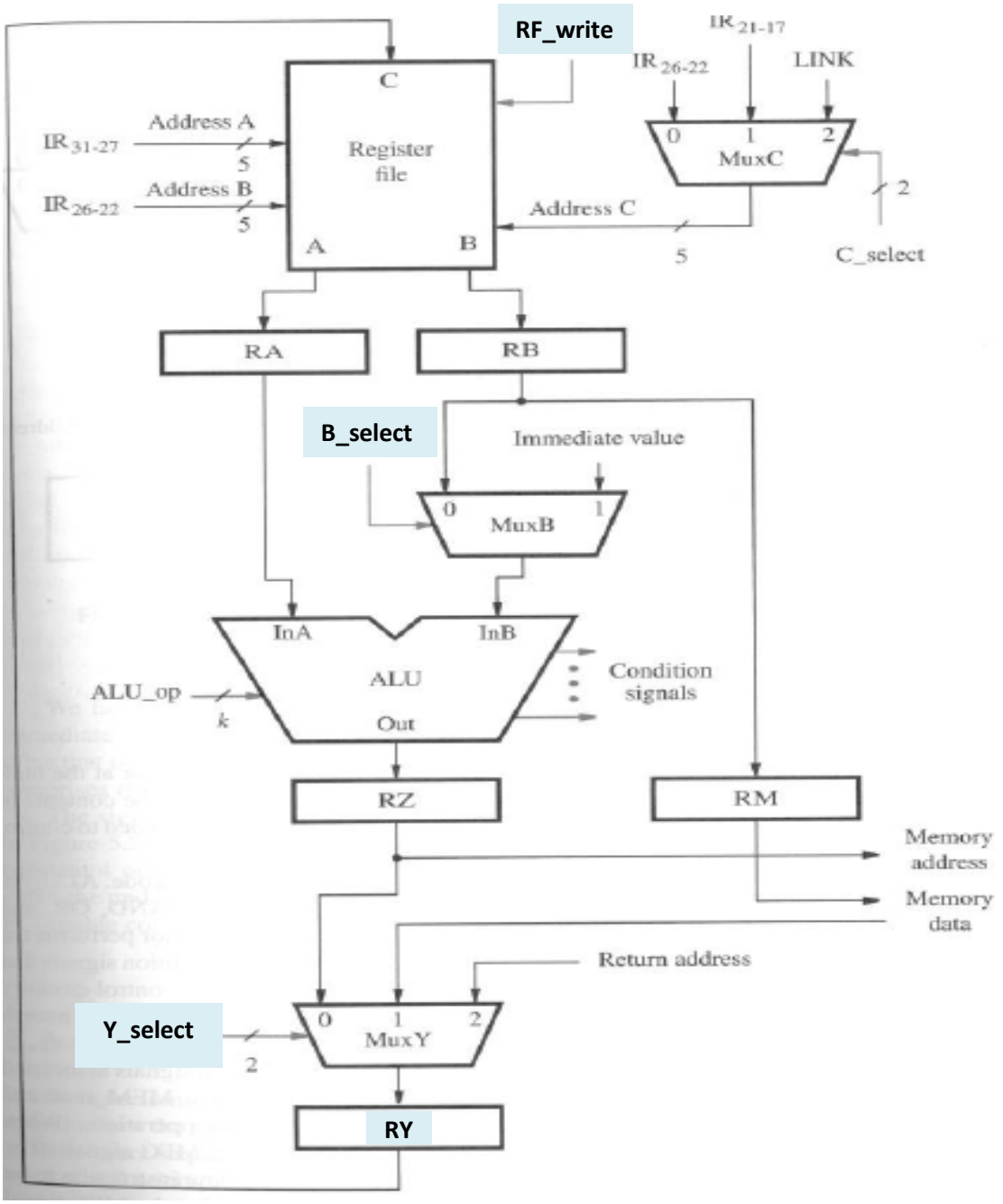


Problem No. 4 (12 points)

Consider the following instruction sequence being processed on the non-pipelined 5-stage RISC processor discussed in class:

Instruction 1: Load R4, #100(R2)
Instruction 2: Add R5, R3, R4
Instruction 3: Store R5, #200(R2)

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



(a) **(8 points)** Write down the values of following control signals:

I. *B_select* during stage 3 of instruction processing (for each of the three instructions).
II. *Y_select* during stage-4 of instruction processing (for instructions 1 and 2 only).
III. *RF_write* during stage-5 of instruction processing (for each of the three instructions).

Solution:

i. *B_select* has values of **1**, **0** and **1** for instructions 1, 2, and 3, respectively.
ii. *Y_select* has values of **1** and **0** for instructions 1 and 2, respectively.
iii. *RF_write* has values of **1**, **1** and **0** for instructions 1, 2, and 3, respectively.

(b) **(4 points)** Assume that the initial contents of registers R2, R3 and R4 are 5000, 200 and 300, respectively. Also assume that the initial contents of memory address 5100 are 400. Write down the contents of inter-stage register *RY* after the completion of stage-4 for instructions 1 and 2.

Solution:

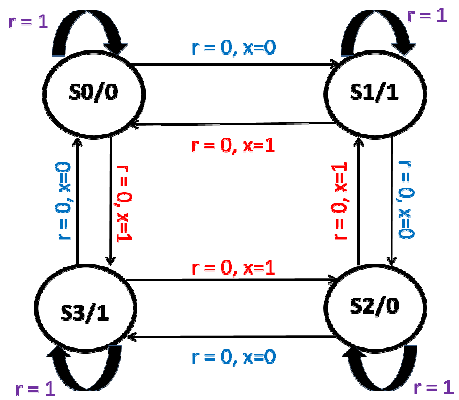
For instruction 1:
Contents of *RY* = Contents of memory address (5000 + 100) = **400**
The contents of register R4 become 400 after instruction 1.

For instruction 2:
Contents of *RY* = [R3] + [R4] = 200 + 400 = **600**

Problem No. 5 (18 points)

(a) **(6 points)** You are required to design a 2-bit synchronous counter by using a finite state machine. The counter has two external input signals *r* and *x*, which dictate the operation of the counter as follows: (i) If ***r* = 0**, ***x* = 0**, the counter counts *up*, (ii) If ***r* = 0**, ***x* = 1**, the counter counts *down*, (iii) If ***r* = 1**, the counter stops counting and retains its present value, irrespective of the value of *x*. 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:

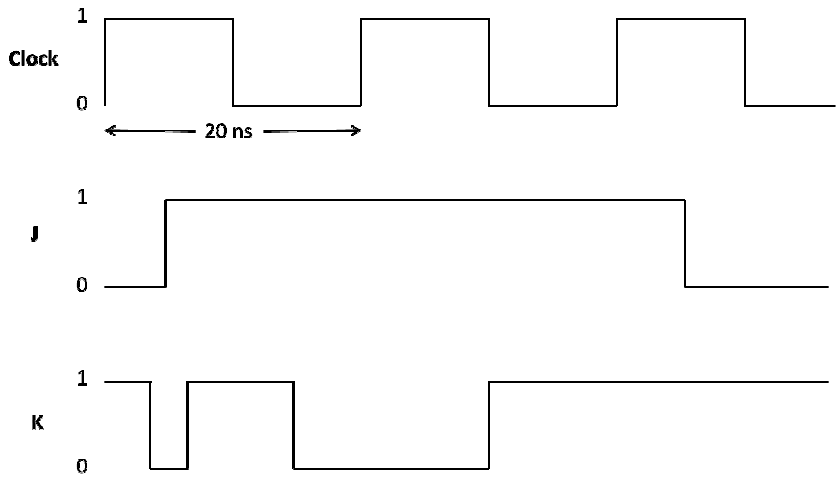


(b) **(6 points)** A 4-bit **carry-lookahead** adder (CLA) is used to add two numbers X ($x_3x_2x_1x_0$) and Y ($y_3y_2y_1y_0$) with a external carry-in of c_0. Recall that the CLA computes the propagate functions ($P_3P_1P_2P_0$) and the generate functions ($G_3G_2G_1G_0$) at each bit position, and then uses them to compute the carry-out values ($c_4c_3c_2c_1$) at each bit position.
Show the logic expressions for the following outputs: (i) P_3, (ii) G_3, and (iii) c_4.

Solution:

i. $P_3 = x_3 + y_3$
ii. $G_3 = x_3.y_3$
iii. $c_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0c_0$

(c) **(6 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) 10 ns, (ii) 20 ns, (iii) 40 ns?



Solution:

i. At t = 10 ns, **Q = 0** (Q retains its initial value until there is a positive clock edge)
ii. At t = 20 ns, **Q = 1** (Positive clock edge arrives at t = 20 ns and J= 1, K = 0)
iii. At t = 40 ns, **Q = 0** (Q toggles its value because J = 1, K = 1 and there is a positive clock edge at 40 ns)

Problem No. 6 (12 points)

A 5-stage pipelined RISC processor C_1 running at 2.2 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 with a prediction accuracy of 80%. The computation of actual branch outcomes is carried out in the "decode" stage. Also assume that C_1 uses a cache such that **100%** of the instruction fetches and ***x* %** of the data accesses hit in the cache. The penalty to access the main memory for a cache miss is 10 cycles.
A customer requires that the processor C_1 must achieve a throughput of 2 billion instructions per second. Calculate the minimum value of x (data hit rate in the cache) which satisfies this requirement?

Solution:

Clock Rate "R" = 2.2 GHz = $2.2*10^9$ 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 hit in the cache, the only source of cache misses are the data accesses caused by Load and Store instructions.
Percentage of Load and Store instructions = 20% + 10% = 30%
Percentage of data accesses that hit in the cache = x%
Therefore, percentage of data accesses that miss in the cache = (100 - x) %
Cache miss penalty = 10 cycles
Therefore, δ_{miss} = stall frequency * stall penalty = 0.3 * ((100-x)/100) * 10 = 3 – 0.03x

For branch mispredictions:
Percentage of branch instructions = 20%
Since branch computation is carried out in "Decode" stage (stage-2), stall penalty = 1 cycle
Branch prediction accuracy = 80%
Therefore, percentage of branches that are predicted inaccurately = 20%
$\delta_{branch\_penalty}$ = stall frequency * stall penalty = (0.2)(0.2) * 1 = 0.04

Adding the two different sources of stall cycles:
Total stall cycles per instruction $\delta = \delta_{miss} + \delta_{branch\_penalty}$ = 3 – 0.03x + 0.04 = 3.04 – 0.03x

Throughput "P_p" = R / (1+δ) = $(2.2*10^9)/(1 + 3.04 - 0.03x) = (2.2*10^9)/(4.04 – 0.03x)$
Since the required throughput is 2 billion instructions per second:
$2*10^9 = (2.2*10^9)/(4.04 – 0.03x)$
4.04 – 0.03x = 2.2 / 2 = 1.1
Solving for "x" in the above equation yields x = 98.
Therefore, the required minimum cache hit rate for data accesses is **98 %**

Problem No. 7 (18 points)

(a) **(6 points)** A computer system uses 32-bit memory addresses. It has a 128K-byte 8-way set-associative cache, with 64 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 = 64 bytes = 2^6 bytes = 2^6 words
Therefore, **Number of bits in the *Word* field = 6**
Cache size = 128 K-byte = 2^{17} bytes
Number of cache blocks per set = 8 = 2^3
Number of sets = Cache size / (Block size * Number of blocks per set) = $2^{17}/(2^6 * 2^3) = 2^8$
Therefore, **Number of bits in the *Set* field = 8**
Total number of address bits = 32
Therefore, **Number of bits in the *Tag* field =** 32 - 6 - 8 **= 18**

(b) **(12 points)** A processor has a small direct-mapped cache capable of holding four cache blocks. Each cache block consists of 32 words. The processor uses 12-bit memory addresses. Assume that the initial tag values for each cache block are as follows:

Block	Tag
00	00110
01	00001
10	00000
11	Invalid

The processor reads data sequentially from the following addresses: 32, 48, 64, 128

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

Solution:

Block size = 32 words = 2^5 words => No. of bits in the *Word* field = 5
Number of cache blocks = 4 = 2^2 => No. of bits in the *Block* field = 2
Total number of address bits = 12 => No. of bits in the *Tag* field = 12 – 5 - 2 = 5
For a given 12-bit address, the 5 most significant bits represent the *Tag*, the next 2 bits represent the *Block*, and the 5 least significant bits represent the *Word*.

Access # 1:
Address = $(32)_{10}$ = $(000000100000)_2$. For this address, *Tag* = **00000**, *Block* = 01
Current tag for cache block 01 = **00001** (initial tag value from table)
Address tag does not match the block tag => cache **miss**
After this access, *Tag* field for cache block 01 is set to 00000

Access # 2:
Address = $(48)_{10}$ = $(000000110000)_2$. For this address, *Tag* = **00000**, *Block* = 01
Current tag for cache block 01 = **00000** (set in access # 1)
Address tag matches the block tag => cache **hit**

Access # 3:

Address = $(64)_{10}$ = $(000001000000)_2$. For this address, *Tag* = **00000**, *Block* = 10

Current tag for cache block 10 = **00000** (initial tag value from table)

Address tag matches the block tag => cache **hit**

Access # 4:

Address = $(128)_{10}$ = $(000010000000)_2$. For this address, *Tag* = **00001**, *Block* = 00

Current tag for cache block 00 = **00110** (initial tag value from table)

Address tag does not match the block tag => cache **miss**