ECE 341 Final Exam Solution

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.

(a) SRAM is more expensive on a cost-per-bit basis as compared to DRAM. **TRUE**
(b) The main purpose of using larger cache block sizes is to reduce the cache access time. **FALSE**
(c) A write-back cache typically requires more reads to the main memory than a write-through cache. **FALSE**
(d) A page table acts as a cache for the Translation Lookaside Buffer (TLB). **FALSE**
(e) Booth algorithm is most efficient when the multiplier has an alternating sequence of 1s and 0s. **FALSE**

Problem No. 2 (10 points)

Multiple possible answers are provided for each of the following questions. Mark all the correct answers for each question.

(a) **(3 points)** Consider a 8M x 128 memory. How many address bits are needed to access the memory?
   i. 23
   ii. 30
   iii. 40
   iv. 7

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

(c) **(3 points)** 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 most significant 34 bits
   ii. The least significant 32 bits
   iii. The most significant 32 bits
   iv. The least significant 12 bits

(d) **(Extra credit question: 4 points)** A processor uses a 3-level cache hierarchy. When a program *P* is run on the processor, the miss ratios for the L1, L2 and L3 caches are 95%, 80% and 60% respectively. What percentage of L1 cache misses hit in the L3 cache?

i. 60%
ii. 32%
iii. 48%

Problem No. 3 (18 points)

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

```
ADD     R2, R3, R4          // Add the contents of registers R3 and R4, write the result into R2
STORE   R5, #200(R2)        // Store the contents of register R5 at memory address 200+[R2]
LOAD    R6, #400(R2)        // Load the contents of memory address 200+[R2] into register R6
BEQ     R5, R6, #8          // Compare the contents of registers R5 and R6; if equal, take the branch
```

Assume that for the "BEQ" instruction in the above sequence, the computation of branch outcomes (comparison of source registers) is carried out in the "Compute" stage (stage-3).

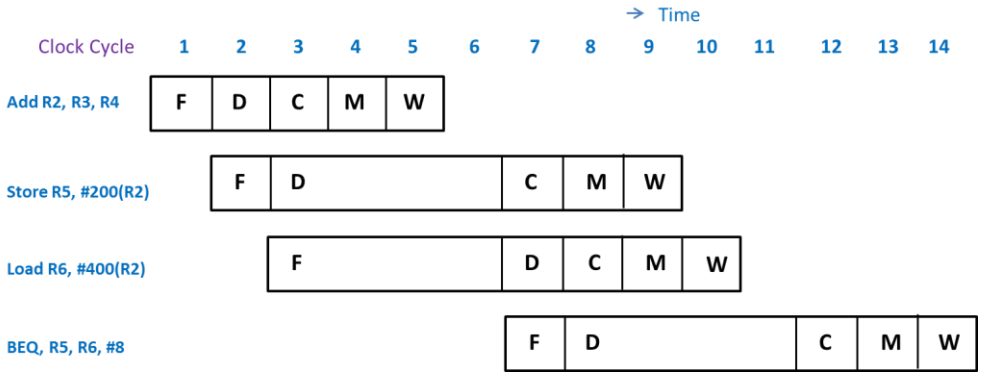
(a) **(6 points)** Identify all the data dependencies in the above instruction sequence. For each dependency, indicate the two instructions and the register that causes the dependency.

Solution:

There are three data dependencies in this instruction sequence:
(i) Store instruction depends on Add instruction for register R2
(ii) Load instruction depends on Add instruction for register R2
(iii) BEQ instruction depends on Load instruction for register R6

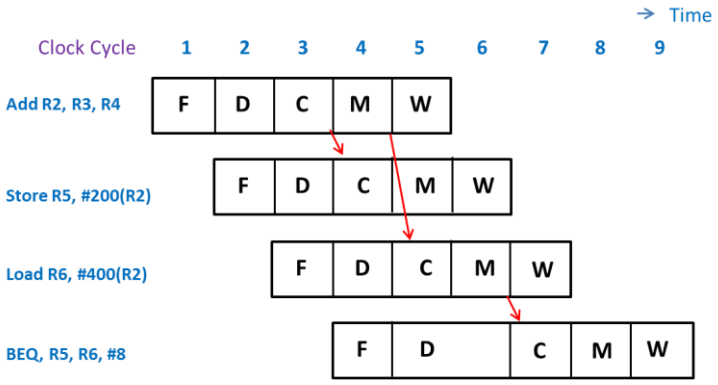
(b) **(6 points)** Assume that the pipeline does not use operand forwarding. Also assume that the only sources of pipeline stalls are the data hazards. Draw a diagram that represents instruction flow through the pipeline during each clock cycle. How long does it take for the instruction sequence to complete?

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:

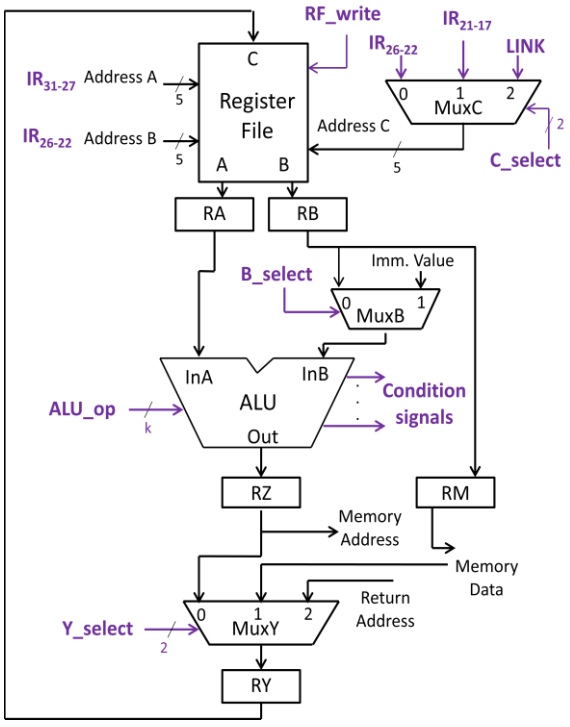


Problem No. 4 (16 points)

(a) **(9 points)** The 5-stage RISC processor discussed in class is used to execute the following sequence of instructions, one after the other:

Instruction-1: LOAD R4, #200(R2)
Instruction-2: AND R5, R3, R4
Instruction-3: CALL_REGISTER R5

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



Fill the following table to indicate the values of control signals for each of the three instructions:

	Y-select in stage-4	*RF_write* in stage-5	*C_select* in stage-5
LOAD	01	1	00
AND	00	1	01
CALL_REGISTER	10	1	10

(b) **(7 points)** Assume that the initial contents of registers R2, R3 and R4 are 2000, 60 and 100, respectively. Also assume that the initial contents of memory addresses 2000 and 2200 are 10 and 26, 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 the program counter (PC) after the completion of above instruction sequence.

Solution:

Instruction-1 (Load): Contents of *RZ* = [R2] + 200 = 2000 + 200 = **2200.** Contents of memory address 2200 (26) are loaded into R4. Therefore R4 contains 26 after instruction-1.

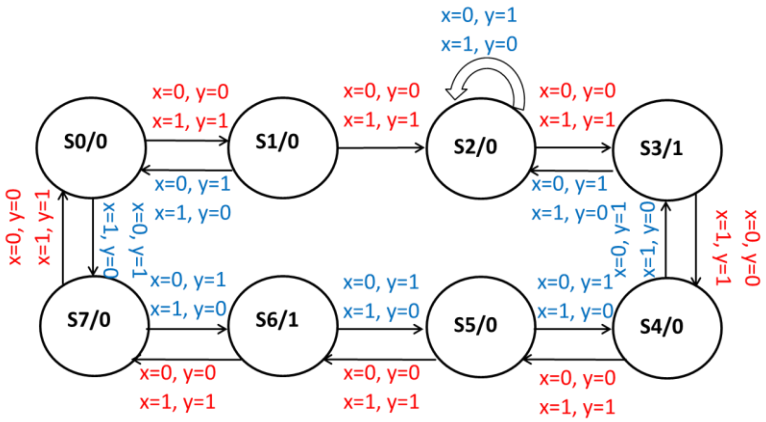
Instruction-2 (And): Contents of *RZ* = [R3] AND [R4] = (60) AND (26) = $(00111100)_2$ AND $(00011010)_2$ = $(00011000)_2$ = **24.** Therefore R5 contains 24 after instruction-2.

Instruction-3 (Call_Register): Contents of R5 loaded into PC. Therefore PC becomes **24**.

Problem No. 5 (14 points)

(a) **(8 points)** You are required to design a 3-bit synchronous counter by using a finite state machine. The counter has two external input signals *x* and *y*, which dictate the operation of the counter as follows: (i) If the two inputs have identical values, the counter counts *up*, (ii) If exactly one of the two inputs is a "0", the counter keeps counting down until it reaches "2" and then stops there. The counter also has an output signal *z*, which is equal to 1 only if the present value of the counter is a non-zero multiple of 3. Draw the state diagram for this state machine.

Solution:



(b) **(6 points)** Prove that the associative rule does not apply to the NOR operator: $(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$

Solution:

We can use the truth table to solve this problem:

x	y	z	x ↓ y	y ↓ z	(x ↓ y) ↓ z	x ↓ (y ↓ z)
0	0	0	1	1	0	0
0	0	1	1	0	0	1
0	1	0	0	0	1	1
0	1	1	0	0	0	1
1	0	0	0	1	1	0
1	0	1	0	0	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	0

By comparing the last two columns of the truth table, we can observe that the two functions are not equivalent. Therefore, the associative rule does not apply to the NOR operator.

Problem No. 6 (14 points)

A 5-stage pipelined RISC processor C_1 running at 2.6 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 to predict the branch instructions with a prediction accuracy of 85%. 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 98**%** of the data reads hit in the cache.
To execute the program completely, the processor C_1 needs to run 20 billion instructions. A customer requires that the program must be completed in less than 10 seconds. What is the upper bound on the main memory latency penalty (in terms of processor cycles) to satisfy the customer requirement?

Solution:

Clock Rate "R" = 2.6 GHz = $2.6*10^9$ 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 = 98%
Therefore, percentage of data accesses that miss in the cache = (100 - 98) % = 2%
Cache miss penalty = M (unknown)
Therefore, δ_{miss} = stall frequency * stall penalty = 0.2 * 0.02 * M = 0.004M

For branch mispredictions:
Percentage of branch instructions = 20%
Since branch computation is carried out in "Compute" stage (stage-3), stall penalty = 2 cycles
Branch prediction accuracy = 85%
Percentage of branches that are predicted inaccurately = (100 - 85) % = 15%
$\delta_{branch\_penalty}$ = stall frequency * stall penalty = 0.2 * 0.15 * 2 = 0.06

Adding the two different sources of stall cycles:
Total stall cycles per instruction $\delta = \delta_{miss} + \delta_{branch\_penalty}$ = 0.004M + 0.06

Throughput "P_p" = R / (1+δ) = $(2.6*10^9)/(1 + 0.004M + 0.06) = (2.6*10^9)/(1.06 + 0.004M)$
Required throughput = 20 billion instructions / 10 seconds = $2 * 10^9$ instructions/second
To satisfy the performance requirement:
$(2.6*10^9)/(1.06 + 0.004M) >= 2*10^9$
Solving for "M" in the above equation yields M <= 60
Therefore, the upper bound on the main memory latency penalty is **60** cycles.

Problem No. 7 (18 points)

(a) **(6 points)** A computer system uses 40-bit memory addresses. It has a 8K-byte fully 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**
Since this is a fully-associative cache, there is only one set in the cache and the **Number of bits in the *Set* field = 0**
Total number of address bits = 40
Therefore, **Number of bits in the *Tag* field** = 40 - 6 = **34**

(b) **(12 points)** A processor that uses 8-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	010	Invalid	000	Invalid

The processor reads data sequentially from the following decimal addresses: 52, 64, 80, 16, 48
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 = 2 (shown in the figure)
Therefore, Number of bits in the *Set* field = 1
Total number of address bits = 8
Therefore, Number of bits in the *Tag* field = 8 - 4 - 1 = 3
For a given 8-bit address, the 3 most significant bits represent the *Tag*, the next bit represents the *Set*, and the 4 least significant bits represent the *Word*.

Access # 1:
Address = $(52)_{10}$ = $(00110100)_2$. For this address, *Tag* = 001, *Set* = 1
Neither of the tags in set 1 match the address tag => cache **miss**
After this access, Tag field for way-1 in set 1 is set to 001

Set	0		1	
Block	0	1	0	1
Tag	010	Invalid	000	001

Access # 2:
Address = $(64)_{10}$ = $(01000000)_2$. For this address, *Tag* = 010, *Set* = 0
Tag for way-0 in set 0 matches the address tag => cache **hit**

Access # 3:
Address = $(80)_{10}$ = $(01010000)_2$. For this address, *Tag* = 010, *Set* = 1
Neither of the tags in set 1 match the address tag => cache **miss**
The LRU replacement algorithm will evict the block in way-0 (the LRU block). Therefore, the Tag field for way-0 in set 1 is set to 010

Set	0		1	
Block	0	1	0	1
Tag	010	Invalid	010	001

Access # 4:
Address = $(16)_{10}$ = $(00010000)_2$. For this address, *Tag* = 000, *Set* = 1
Neither of the tags in set 1 match the address tag => cache **miss**
The LRU replacement algorithm will evict the block in way-1 (the LRU block). Therefore, the Tag field for way-1 in set 1 is set to 000

Set	0		1	
Block	0	1	0	1
Tag	010	Invalid	010	000

Access # 5:
Address = $(48)_{10}$ = $(00110000)_2$. For this address, *Tag* = 001, *Set* = 1
Neither of the tags in set 1 match the address tag => cache **miss**