CSE/ECE 511: Computer Architecture
End Semester Examination Rubric

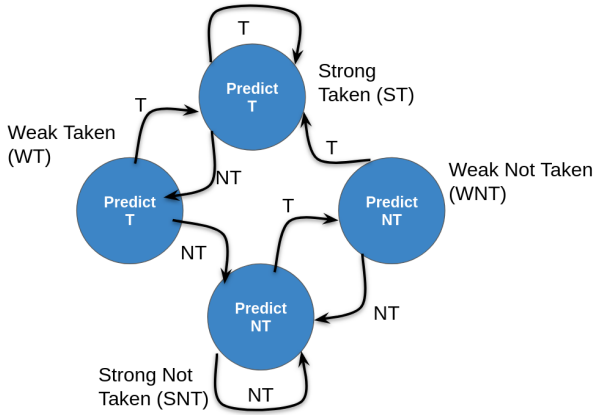
Q1. Consider the following pseudo-code snippet with line numbers: **[18 Marks]**

```
1.          i = 0;
2.  label:  i ++;
3.          while (i < 5)
4.                  {
5.                          if (i mod 2 == 0) then
6.                                  jump label ;
7.                          i++;
8.                  }
```

(a) Consider a 1-bit branch predictor FSM with states taken (T) and not-taken (NT). This predictor is shared by all the branches to be taken in above code. Fill the following table when the above code is executed. Assume the 1-bit branch predictor has not-taken (NT) as the initial state.

Predicted/Previous State	Actual/Updated State	Misprediction yes/no?

(b) Repeat part (a) for 2-bit branch predictor with FSM shown in the following figure, with states shown. The rest of the conditions remain the same.



Note: *If the condition in parentheses is found to be true, then the branch is assumed to be taken*

Solution:

<u>Note for partial marking</u>: If only one branch has been considered, maximum marks are capped at 50% of the total marks for that question and the part. The exact marks obtained depend on the overall correctness of the answer and if the concept is applied correctly.

(a) <u>1-bit BP</u>: 1*9 = 9 Marks

S.No.	Info	Predicted/Previous State (NT/T)	Actual/Updated State (NT/T)	Misprediction yes/no?
1	i=1	NT	T	Y
2		T	NT	Y
3	i=2	NT	T	Y
4		T	T	N
5	i=3	T	T	N
6		T	NT	Y
7	i=4	NT	T	Y
8		T	T	N
9	i=5	T	NT	Y

(b) 2-bit BP: 1*9 = 9 Marks

S.No.	Info	Predicted/Previous State (NT/T)	Actual/Updated State (NT/T)	Misprediction yes/no?
1	while; i=1	SNT	WNT	Y
2	if; i=1	WNT	SNT	N
3	while; i=2	SNT	WNT	Y
4	if; i=2	WNT	ST	Y
5	while; i=3	ST	ST	N
6	if; i=3	ST	WT	Y
7	while; i=4	WT	ST	N
8	if; i=4	ST	ST	N
9	while; i=5	ST	WT	Y

Q2. Consider that program P1 is stored in the RAM with a base address of 2048 and a bound address of 3072. Convert the addresses used in the program to physical addresses. Indicate when bounds violations occur in the program. **[10 Marks]**

Instruction Address (Virtual)	Instruction
0	add R1, R2, R3
4	lw R2, 100 (R3)
8	sw R3, 1000 (R4)
12	sub R4, R5, R6
16	sw R4, 60 (R7)
20	add R5, R4, R7
24	sw R5, 1032 (R6)

Ans:
2048: add R1, R2, R3
2052: lw R2, 2148 (R3)
2056: sw R3, 3048 (R4)
2060: sub R4, R5, R6
2064: sw R4, 2108 (R7)
2068: add R5, R4, R7
2072: sw R5, 3080 (R6) ->Bounds violation (not executed)
(1 + 1.5 + 1.5 + 1 + 1.5 + 1 + 1.5 + 1 = 10 Marks)

Q3. Assume a 32-bit, byte-addressed machine with virtual addressing. However, any memory address whose three high-order bits are 111 is treated as unmapped. These addresses are only accessible in privileged mode—i.e., by the operating system—and bypass virtual address translation. **[10 Marks]**

a. What is the maximum amount of physical memory this system can address? **[1 Mark]**
b. What is the maximum amount of virtual memory any single user process/program on this system can address? **[1 Mark]**
c. What is the maximum number of virtual pages that are available to each process/program, assuming the page size is 32KB? **[2 Marks]**
d. Assuming each page table entry is 4 bytes, what is the size of the page table required for a program with a maximum number of virtual pages? **[1.5 Marks]**
e. Repeat parts c and d for a page size of 64 KB. **[3.5 Marks]**
f. What disadvantages does a large page size have? **[1 Mark]**

Ans. ***<u>Note:</u> Marks will not be awarded if the unit such as byte/bits/GB/MB/KB is not specifiied or used wrongly in the answer.***

a. For byte-addressable machines, 2^(32) bytes = 4 GB can be addressed.
b. Given virtual address translation does not happen when higher order bits are 111, the maximum address space is 2^32 - 2^29 = 7*2^9 = 3.5 GB.
c. Maximum number of virtual pages = (Maximum virtual address size)/ (Page size)
   = (7*2^29)/32 KB
   = (7*2^29)/2^15 = 7*(2^14) = 114688
d. Page table size = 7*(2^14)* 4 = 7*(2^16) = 448 KB
e. Maximum virtual pages = 57344, page table size = 224 KB
f. Keywords: Wasted Memory, high page fault

Q4. Consider a processor with a cycle time of 0.4ns. Assume 3 caches, Direct mapped, 2-way set associative, and 4-way set-associative caches. All caches have the same size of 8KB. Miss rates of the three caches are 6.8%, 4.9%, and 4.4% respectively. Assume the cache miss penalty to be the same for all types, i.e., 100ns. You might be aware that introducing associativity increases the hardware complexity in terms of MUXs. As a result, for associative caches(Any type), the cycle time is increased to 1.4ns. Assume a Hit-time of 1 cycle. Find the AMAT(Average memory access time) for all three configurations. **[2*3 = 6 Marks]**

Ans.

For direct mapped cache, AMAT = 1*0.4 + (6.8/100)*100 = 7.2 ns
For 2 way set associative cache, AMAT = 1*1.4 + (4.9/100)*100 = 6.3 ns
For 4 way set associative cache, AMAT = 1*1.4 + (4.4/100)*100 = 5.8 ns

Q5. Consider the I2O2 and I2OI type of processor implementations. The pipeline has 4 functional units: ALU (1 cycle, X0), Loads and stores (2 cycles, S0 and S1), and multiply (4 cycles, Y0, Y1, Y2, Y3). Draw the pipeline diagrams of the following set of assembly instructions for both implementations and report the CPI for both cases. **[12 Marks]**

```
ADD   R1   R2   R3
SW    R5   10(R1)
MUL   R1   R2   R5
LD    R6   10(R8)
```

Solution:

<u>Note for partial marking</u>: For wrongly attempted pipeline, maximum marks are capped at 50% of the total marks of that question and the part. The exact marks obtained depend on the overall correctness of the answer and if the concept is applied correctly.

<u>Full-Bypassing:</u>

Q1: I2O2	**Full Bypassing**												
Cycles -->	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
ADD R1 R2 R3	F	D	I	X0	W								
SW R5 10(R1)		F	D	I	M0	M1	W						
MUL R1 R2 R5			F	D	I	Y0	Y1	Y2	Y3	W			
LD R6 10(R8)				F	D	I	M0	M1	W				
Q1: I2OI													
Cycles -->	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
ADD R1 R2 R3	F	D	I	X0	W	C							
SW R5 10(R1)		F	D	I	M0	M1	W	C					
MUL R1 R2 R5			F	D	I	Y0	Y1	Y2	Y3	W	C		
LD R6 10(R8)				F	D	I	M0	M1	W	r		C	

I2O2: 0.5+1+2+2 [5.5 Marks]
I2OI: 0.5+1+2+2 [5.5 Marks]
CPI_I2O2: 10/4 = 2.5 [0.5 Marks]
CPI_I2OI: 12/4 = 3 [0.5 Marks]

No-Bypassing:

Q1: I2O2	**No Bypassing**														
Cycles -->	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**		
ADD R1 R2 R3	F	D	I	X0	W										
SW R5 10(R1)		F	D	I	I	I	M0	M1	W						
MUL R1 R2 R5			F	D	D	D	I	Y0	Y1	Y2	Y3	W			
LD R6 10(R8)				F	F	F	D	I	M0	M1	W				
Q1: I2OI															
Cycles -->	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
ADD R1 R2 R3	F	D	I	X0	W	C									
SW R5 10(R1)		F	D	I	I	I	I	M0	M1	W	C				
MUL R1 R2 R5			F	D	D	D	D	I	Y0	Y1	Y2	Y3	W	C	
LD R6 10(R8)				F	F	F	F	D	I	M0	M1	W	r		C

I2O2: 0.5+1+2+2 **[5.5 Marks]**
I2OI: 0.5+1+2+2 **[5.5 Marks]**
CPI_I2O2: 12/4 = 3 **[0.5 Marks]**
CPI_I2OI:15/4 = 3.75 **[0.5 Marks]**

Q6. Consider a VLIW EQ compiler with the processor having two integer units and one load/store unit. *add* and *sub* take one clock cycle, *mul* (integer multiplication) takes two clock cycles, and *lw* and *sw* take three clock cycles. Assume the compiler does not reorder the instructions. Pack the instructions for the program below. **[10 Marks]**

add R1, R2, R3
sub R2, R3, R4
mul R4, R2, R6
lw R3, 4(R6)
sw R5, 8(R4)
add R1, R5, R3
sub R7, R9, R5
add R10, R7, R8

Sample table for this question:

I1	I2	Load/Store

Ans.

I1	I2	Load/Store
add R1, R2, R3	sub R2, R3, R4	
mul R4, R2, R6		lw R3, 4(R6)
add R1, R5, R3	sub R7, R9, R5	sw R5, 8(R4)
	add R10, R7, R8	

(1+1+ 1.5 + 1.5 + 1.5+1+1+1.5) Marks

Q7. Consider the usage of critical word first and early restart on L2 cache misses. If the processor wants to read a word 0x12AB and occur a miss in L2, the fetching in a conventional processor without any optimization is done from the main memory in the following sequences: CB12,12AB, 47A1, 34CD. Assuming a 1MB L2 cache with 64 bytes blocks and a refill path (i.e bus from memory) that is 16 bytes wide. Assume that the L2 can be written with 16 bytes every 4 processor cycles; the time to receive the first 16-byte block from the memory controller is 120 cycles, and each additional 16-byte block from the main memory requires 16 cycles and data can be bypassed directly into the read port of the L2 cache. Ignore any cycles to transfer the miss request to the L2 cache, and the requested data to the L1 cache.

a) How many cycles would it take to service an L2 cache miss
   i) Without critical word first and early restart?
   ii) With only an early restart?
   iii) With critical word first and early restart?

b) Is critical word first and early restart better to implement in L1 cache or L2 caches? Give a reason for your answer. **[2x3+1+2 = 9 Marks]**

Ans:

a)

i) Without critical word first and early restart

L2 block size = 64/16 = 4 word

The number of cycles = Cycle for 1st word + 3 x Cycle for the remaining word

= 120 + 3 x 16 = 168

ii) The number of cycles with only an early restart = 120 + 16 = 136

iii) The number of cycles with critical word first and early restart = 120

b) L2 implementation is better.

Reason: The benefits of critical word first and early restart **depend on the block size**. Since, **L2 has a larger cache block size**. Thus, it's more important to L2, which do not have to wait the whole block. While L1 block is generally smaller, the improvement may not be significant. It also depends on the average memory access time and percent reduction in missed service times

Q8.

a. How are Hexagon DSPs in Snapdragon™ 820 architecture able to achieve low power consumption compared to CPU? **[2 Marks]**

b. Why is Heaxogon DSP connected directly to the L2 cache? **[1 Mark]**

Ans:

a. Unlike CPU, the bulk of the power in DSP is used in **compute datapath.** Also, it used an in-order **VLIW architecture with long vectors**, which amortized the overhead of data movement. In addition, it works at a lower **clock cycle**, reducing overheads and leakage.

   Any two keywords are needed.

b. A large amount of data needed for Hexagon DSP can be stored only in the L2 cache.