<u>SC3050 Examination 2020-2021 Semester 2</u>

1. (a) The speedup of machine M1 over M2 for a given program is 2. Let machines P1 and P2 be the enhanced versions of M1 and M2, respectively. It is found that one-third of the program runs 2 times faster in P1 than in M1 and half of the program runs 3 times faster in P2 than in M2. Calculate the speedup of P1 over P2.

(6 marks)

Answer

T1 of P1 on M1 $= \frac{1}{3} * \frac{\text{Time}_{M1}}{2} = \frac{1}{6}\text{Time}_{M1}$

T2 of P1 on M1 $= \frac{2}{3}\text{Time}_{M1}$

Total T of P1 $= \frac{1}{6}\text{Time}_{M1} + \frac{2}{3}\text{Time}_{M1} = \frac{5}{6}\text{Time}_{M1}$

T1 of P2 on M2 $= \frac{1}{2} * \frac{\text{Time}_{M2}}{3} = \frac{1}{6}\text{Time}_{M2}$

T2 of P2 on M2 $= \frac{1}{2}\text{Time}_{M2}$

Total T of P2 $= \frac{1}{6}\text{Time}_{M2} + \frac{1}{2}\text{Time}_{M2} = \frac{2}{3}\text{Time}_{M2}$

Speedup of M1 over M2 $= \frac{\text{Time}_{M2}}{\text{Time}_{M1}} = 2$

$\Rightarrow$ $\text{Time}_{M2} = 2(\text{Time}_{M1})$

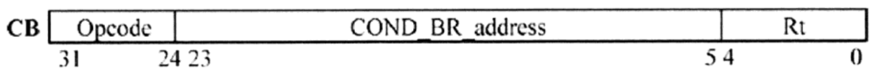
Speedup of P1 over P2 $= \frac{\text{T of P2}}{\text{T of P1}} = \frac{\frac{2}{3}\text{Time}_{M2}}{\frac{5}{6}\text{Time}_{M1}}$

Substitute $\text{Time}_{M2} = 2(\text{Time}_{M1})$

Speedup of P1 over P2 $= \frac{\frac{2}{3}(2)\text{Time}_{M1}}{\frac{5}{6}\text{Time}_{M1}} = \frac{8}{5} = 1.6$

1. (b) State the addressing mode used by the conditional branch instruction "CBNZ X1, address" in the LEGv8 architecture. If the content of the 64-bit program counter (PC) is 0xCC, find the maximum possible address of the instruction memory to which the current conditional branch instruction "CBNZ X1, address" could branch forward.

(7 marks)



Answer

Conditional branch is using PC-relative addressing mode.

No. of address bits for conditional branch = 23 – 5 + 1 = 19 bits

The maximum positive number for 19 bits
= 011 1111 1111 1111 1111 (0x3FFFF)

Maximum PC = PC + Sign extended (offset) << 2
= 0xCC + (0x3FFFF)*4
= 0x0000 0000 0010 00C8

The maximum negative number for 19 bits
= 100 0000 0000 0000 0000 (0x40000)

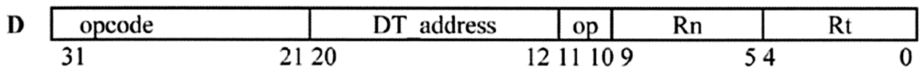
Minimum PC = PC + Sign extended (offset) << 2
= 0xCC + (-0x40000)*4
= 0xFFFF FFFF FFF0 00CC

1. (c) Briefly explain the working of the instructions "LDUR X0, [X1, #8]" and "XOR X2, X1, X3". Use a neat diagram to show the datapath of a single-cycle architecture that supports the execution of both given instructions with minimal number of multiplexers (control signals can be simplified).

(12 marks)

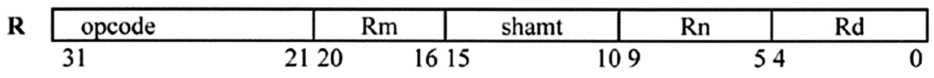
Answer

"LDUR X0, [X1, #8]"

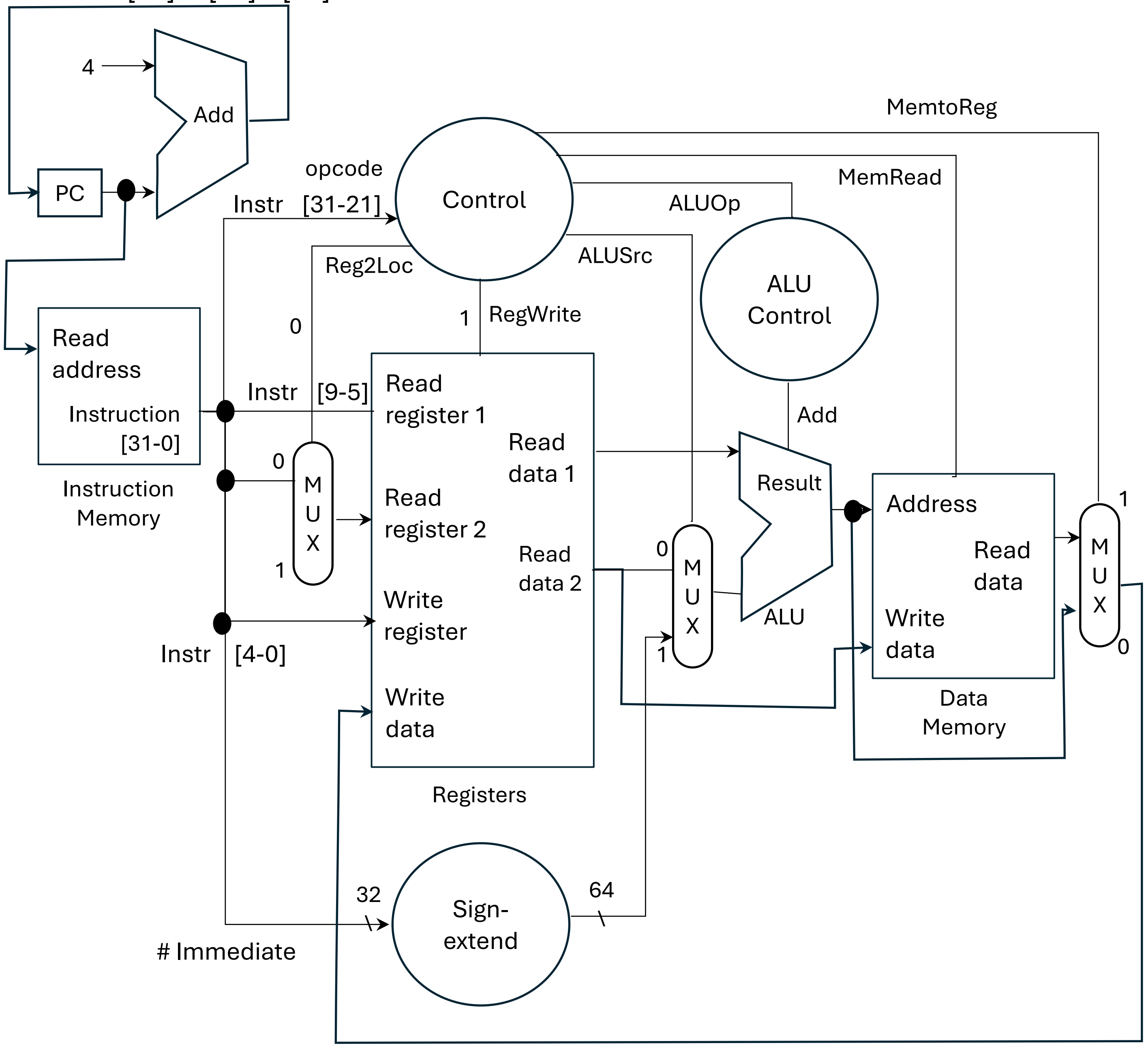


1. Read the data from register X1.
2. Compute the memory address by adding the values of register X1 to the immediate value 8 using the ALU.
3. Write the values from the computed memory address to the register X0 to.
4. [X0] ← mem[[X1] + [8]]

"XOR X2, X1, X3"



1. Read the data from register X1 and register X3.
2. Performs a bitwise XOR operation between values in registers X1 and X3.
3. Write the result in register X2.
4. [X2] ← [X1] ^ [X3]



2. (a) Listing Q2 shows a code segment that is intended to be executed in a 5-stage pipelined LEGv8 processor which can perform write-back and register read operations of different instructions in the same clock cycle. The program counter is updated with the branch target address at the Execute stage. Let the initial values be `X5=0x0000000010000000` and `X6=0x0000000000001000` (`CBNZ`: *branch on not equal to zero*).

<u>Listing Q2</u>

```
I1    loop: LDUR  X0,  [X5,  #0]
I2          LDUR  X1,  [X6,  #0]
I3          ADD   X2,  X1,   X0
I4          XORI  X2,  X2,   #15
I5          STUR  X2,  [X6,  #0]
I6          SUBI  X5,  X5,   #8
I7          SUBI  X6,  X6,   #8
I8          CBNZ  X6,  loop
      finish
```

(i) Calculate the steady-state CPI of the code segment in Listing Q2 with the help of a reservation table for the execution of the code if no data forwarding is allowed. Also find the total number of loop iterations.

(6 marks)

(ii) Perform loop unrolling by a factor of 2 for the code segment in Listing Q2 and do the necessary reordering of instructions to reduce the number of stall cycles to the minimum. Find the steady state CPI achieved by such loop unrolling and instruction reordering when no data forwarding is allowed. You are allowed to use new temporary registers to get rid of hazards.

(6 marks)

(iii) The code segment shown in Listing Q2 is now intended to be executed in a two-way superscalar processor. In the superscalar processor, one way is exclusively for load and store instructions whereas the other way can execute all instructions except load and store. Find the CPI achieved by the superscalar architecture. Note that, no data forwarding is allowed but write-back and register-read operations of different instructions can be performed in the same clock cycle.

(8 marks)

2 (a) (i) Answer

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
loop: LDUR X0, [X5, #0]	I1.	F	D	E	M	W																		
LDUR X1, [X6, #0]	I2.		F	D	E	M	$\boxed{W}$																	
ADD X2, X1, X0	I3.			F	S	S	$\boxed{D}$	E	M	W														
XORI X2, X2, #15	I4.						F	S	S	D	E	M	$\boxed{W}$											
STUR X2, [X6, #0]	I5.									F	S	S	$\boxed{D}$	E	M	W								
SUBI X5, X5, #8	I6.												F	D	E	M	W							
SUBI X6, X6, #8	I7.													F	D	E	M	$\boxed{W}$						
CNBZ X6, loop	I8.														F	S	S	$\boxed{D}$	E	M	W			
finish																		S	S	F	D	E	M	W

$$\text{Steady state CPI} = \frac{\text{No. of instructions + No. of stalls + No. of control}}{\text{No. of instructions}}$$

$$= \frac{8+8+2}{8} = \frac{9}{4} = 2.25$$

X8 = 0x0000000000001000 (hex)
= 4096 (ten)

$$\text{Total number of loop iterations} = \frac{4096}{8} = 512$$

2 (a) (ii) Answer

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
loop: LDUR X0, [X5, #0]	I1.	F	D	E	M	W																		
LDUR X1, [X6, #0]	I2.		F	D	E	M	W																	
ADD X2, X1, X0	I3.			F	S	S	D	E	M	W														
XORI X2, X2, #15	I4.						F	S	S	D	E	M	W											
STUR X2, [X6, #0]	I5.									F	S	S	D	E	M	W								
SUBI X5, X5, #8	I6.												F	D	E	M	W							
SUBI X6, X6, #8	I7.													F	D	E	M	W						
CNBZ X6, loop	I8.														F	S	S	D	E	M	W			
finish																		S	S	F	D	E	M	W

Unrolling by 2

```
loop: LDUR  X0,   [X5,  #0]
      LDUR  X1,   [X6,  #0]
      ADD   X2,   X1,   X0
      XORI  X2,   X2,   #15
      STUR  X2,   [X6,  #0]
      LDUR  X3,   [X8,  #-8]
      LDUR  X4,   [X9,  #-8]
      ADD   X7,   X4,   X3
      XORI  X7,   X7,   #15
      STUR  X7,   [X9,  #-8]
      SUBI  X5,   X5,   #16
      SUBI  X6,   X6,   #16
      CBNZ  X6,   loop
finish
```

Unrolling by 2 and reordering

```
loop: LDUR  X0,   [X5,  #0]
      LDUR  X1,   [X6,  #0]
      LDUR  X3,   [X8,  #-8]
      LDUR  X4,   [X9,  #-8]
      ADD   X2,   X1,   X0
      SUBI  X5,   X5,   #16
      ADD   X7,   X4,   X3
      XORI  X2,   X2,   #15
      XORI  X7,   X7,   #15  (1 stall)
      STUR  X7,   [X9,  #-8] (1 stall)
      SUBI  X6,   X6,   #16
      STUR  X11,  [X10, #-8]
      CBNZ  X6,   loop       (1 stall)
finish                       (2 stalls)
```

$$\text{Steady state CPI} = \frac{\text{No. of instructions} + \text{No. of stalls} + \text{No. of control}}{\text{No. of instructions}}$$

$$= \frac{13 + 3 + 2}{13} = \frac{18}{13} = 1.38$$

2 (a) (iii) Answer

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
loop: LDUR X0, [X5, #0]	I1.	F	D	E	M	W																		
LDUR X1, [X6, #0]	I2.		F	D	E	M	W																	
ADD X2, X1, X0	I3.			F	S	S	D	E	M	W														
XORI X2, X2, #15	I4.						F	S	S	D	E	M	W											
STUR X2, [X6, #0]	I5.									F	S	S	D	E	M	W								
SUBI X5, X5, #8	I6.												F	D	E	M	W							
SUBI X6, X6, #8	I7.													F	D	E	M	W						
CNBZ X6, loop	I8.														F	S	S	D	E	M	W			
finish																		S	S	F	D	E	M	W

	Way-1 (rest of instructions)	Way -2 (LDUR and STUR only)	Cycle
loop	nop	LDUR X0, [X5, #0]	1
	nop	LDUR X1, [X6, #0]	2
	nop	nop	3
	SUBI X5, X5, #8	nop	4
	ADD X2, X1, X0	nop	5
	nop	nop	6
	nop	nop	7
	XORI X2, X2, #15	nop	8
	SUBI X6, X6, #8	nop	9
	nop	nop	10
	nop	STUR X2, [X6, #0]	11
	CNBZ X6, loop	nop	12
	nop	nop	13
	nop	nop	14

CPI after 2-way superscalar architecture

$$= \frac{14}{8} = \frac{7}{4}$$

$= 1.75$

2 (b) Consider the following sequence of actual outcomes for a branch (N N N N N N N, T N T N T N), where T means that the branch is taken and N means that the branch is not taken.

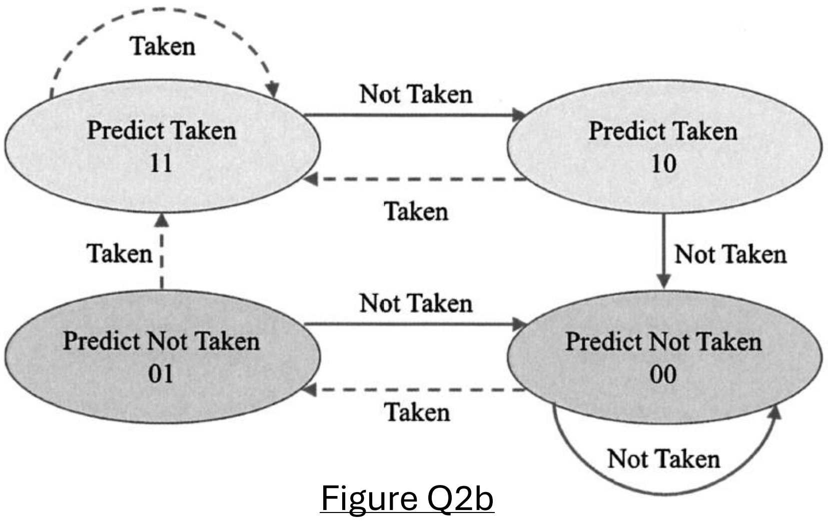


Figure Q2b

Assume that there is only one branch instruction in the program. What is the prediction accuracy for the last 6 occurrences of this branch if the 2-bit branch predictor as shown in Figure Q2b is applied?

(5 marks)

Answer

After the first 7 outcomes are not taken (N), we can assume that the state will be at Predict Not Taken 00.

State	00	01	00	01	00	01
Prediction	N	N	N	N	N	N
Actual	T	N	T	N	T	N
		✓		✓		✓

Prediction accuracy = 3 / 6 = 50%

3. (a) Name three cache replacement algorithms and two write policies for cache.

(5 marks)

Answer

- Cache replacement algorithms
  - FIFO (First-in-first-out)
  - LRU (least recently used)
  - NMRU (not most recently used)
  - Pseudo-random

- Write policies
  - Write-through policy
  - Write-back policy

3. (b) Figure Q3b depicts the memory access workflow for a byte-addressable machine A. Make use of the information in Figure Q3b to answer the following questions.

(i) Given that the L1 cache is a direct-mapped cache which contains 512 cache blocks with the block size of 128 bytes, calculate **Y** and **Z** in Figure Q3b.

(3 marks)

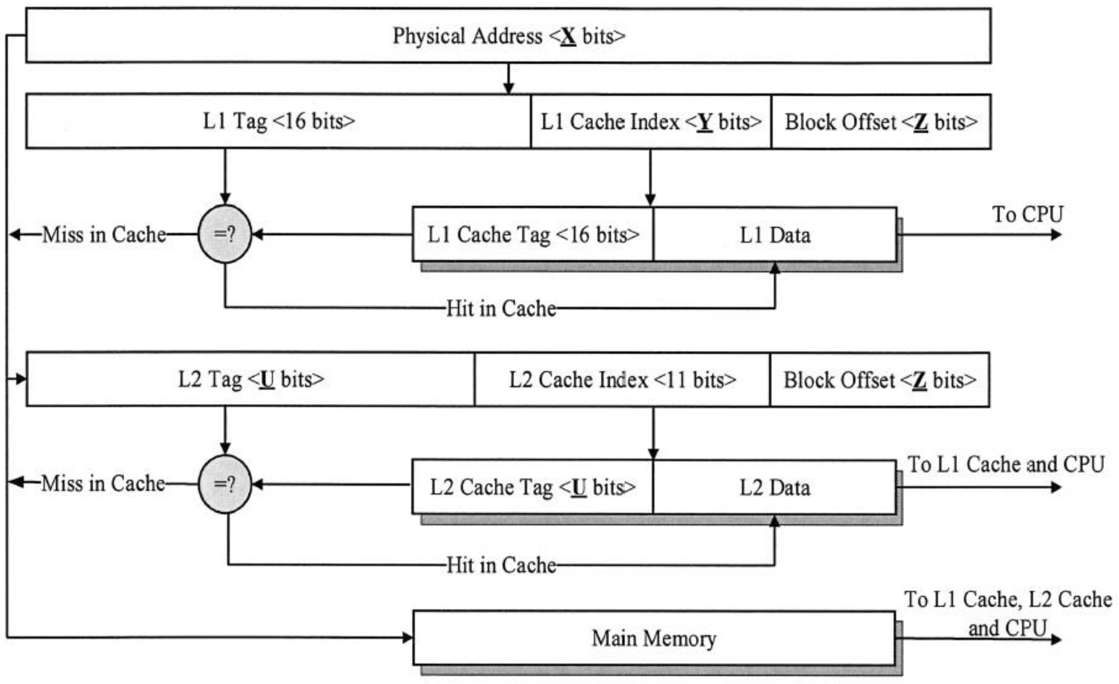


Figure Q3

Answer

No. of block	= 512
No. of way	= 1
Block size	= 128 Bytes
Tag	= 16 bits

No. of set $= \dfrac{\text{No. of block}}{\text{No. of way}} = \dfrac{512}{1} = 512$

Index Y $= \log_2(\text{No. of set}) = \log_2(512) = 9$ bits

Offset Z $= \log_2(\text{Block size}) = \log_2(128) = 7$ bits

3. (b) (ii) What is the size of the main memory of the machine A? Suppose a program accesses the physical memory in a random fashion, that is, random physical addresses are uniformly accessed for a sufficiently long time. What is the miss rate on the L1 cache?

(6 marks)

Answer

Address size = Tag + Index + Offset = 16 + 9 + 7 = 32 bits

Main memory size = 2^{32} = 4GB

No. of block = $\frac{\text{Cache size}}{\text{Block size}}$

Cache size = No. of blocks * Block size = 512 * 128 = 64KB

L1 Hit rate = $\frac{\text{Cache size}}{\text{Memory size}} = \frac{64 * 1024}{4 * 1024 * 1024 * 1024} = \frac{1}{2^{16}}$

L1 Miss rate = 1 – Hit rate = $1 - \frac{1}{2^{16}}$ = 0.99998 = 99.998%

(iii) Given that the L2 cache is a 4-way set associative cache in which a cache entry includes a valid bit, a dirty bit, a use bit and two LRU (Least Recently Used) bits in addition to a tag field and a data field, what is the size of the L2 cache?

(6 marks)

Answer

L2 No. of way = 4
L2 Valid = 1 bit
L2 Dirty = 1 bit
L2 Use = 1 bit
L2 LRU = 2 bits

L2 Index = 11 bits
L2 No. of set = $2^{\text{Index bits}}$ = 2^{11} = 2048 bits

L2 Offset = 7 bits
Tag = Address size – Index – Offset = 32 – 11 – 7 = 14 bits

L2 Block size = 128 Bytes
L2 Data size = Block size = 128 * 8 = 1024 bits

L2 Cache size = (Valid + Dirty + use + LRU + Tag + Data) * Way * Set
= (1 + 1 + 1 + 2 + 14 + 1024) * 4 * 2048
= 8544256 bits
= 1068032 Bytes
= 1043 KB

3. (b) (iv) Suppose the miss rates of the L1 and L2 caches are 5% and 1%, respectively. The access times for the L1 cache, the L2 cache and the main memory are 4, 20 and 100 cycles, respectively. What is the average memory access time (AMAT) of the machine A?

(5 marks)

Answer

L1 Miss rate = 5% = 0.05
L2 Miss rate = 1% = 0.01

L1 Cache = L1 Hit time = 4 cycles
L2 Cache = L2 Hit time = 20 cycles
Main memory = Miss penalty = 100 cycles

AMAT = Hit time + Miss rate * Miss penalty
L1 AMAT = 4 + 0.05 * 100 = 9 cycles
L2 AMAT = 20 + 0.01 * 100 = 21 cycles

L1 + L2 AMAT = L1 Hit time + L1 Miss rate * L2 AMAT
= 4 + 0.05 * 21
= 5.05 cycles

4 (a) Briefly describe how a typical GPU architecture is designed for SIMD/SIMT based parallel programming paradigm.

(5 marks)

Answer

- GPU consists of many SM cores, each with many ‘ALUs’
- Enable massive parallel MAC (floating points) operations
- Warp executes one common instruction for all its threads at a time
- Instructions are pipelined to achieve instruction-level parallelism
- Instructions are issued in order, no branch prediction and speculative execution
- Individual threads are free to branch and execute independently when the thread diverges

4. (b) The code snippet in Figure Q4a shows a program that uses a CUDA kernel `saxpy()` to compute the SAXPY (Single precision A.X plus Y) operation:

$$\mathbf{Y} = A\mathbf{X}+\mathbf{Y}$$

where A is a scalar, while **X** and **Y** are vectors each consisting of `N` floating-point numbers.

```
Line
1   __global__
2   void saxpy(int n, float a, float *x, float *y){
3     int i = blockIdx.x * blockDim.x + threadIdx.x;
4     if (i < n)
5        y[i] = a*x[i] + y[i];
6     }
7
8   int main(void){
9     int N = 4096;          // size of vectors X and Y
10    float A = 3.0;
11    float X[N] = {.....}; // initialize vector X
12    float Y[N] = {.....}; // initialize vector Y
13    int *d_X, *d_Y;
14    cudaMalloc((void**))&d_X, sizeof(float)*N);
15    cudaMalloc((void**))&d_X, sizeof(float)*N);
16    cudaMemcpy(d_X, X, sizeof(float)*N, cudaMemcpyHostToDevice);
17    cudaMemcpy(d_Y, Y, sizeof(float)*N, cudaMemcpyHostToDevice);
18    saxpy<<<.....>>>(.....);
:        :
:     }
```

Figure Q4a

(i) Complete the code shown in Line ***18*** if the number of threads per block is set as 512.

(5 marks)

Answer

No. of threads per block = 512

No. of block $= \dfrac{4096}{512} = 8$

Code = `saxpy<<<8,512>>>(N, A, d_X, d_Y);`

4. (b) (ii) Assume a Stream Multiprocessor (SM) in a GPU has sufficient register and shared memory resources to reside all the blocks. What is the total number of warps that will be created by launching the kernel? Please elaborate with working details.

(4 marks)

Answer

No. of threads per block = 512

No. of warp per block $= \frac{512}{32} = 16$

No. of block = 8

Total no. of warps = No. of warp * No. of block
= 16 * 8
= 128 warps

(c) Figure Q4b shows a CUDA kernel that runs on a GPU to compute the dot product of two vectors A and B and outputs a scalar value C:

$$C = \mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^{N} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_N b_N$$

```
Line
1  void dot_prod(int N, int *a, int *b, int *c){
2    int temp [N];
3    int i = threadIdx.x;
4    temp [i] = a[i]*b[i];
5
6    // Thread 0 sums the pairwise products
7    if (i == 0) {
8       int sum = 0;
9       for (int j = 0; j < N; j++)
10        sum += temp [j];
11      *c = sum;
12   }
13 }
```

Figure Q4b

4. (c) (i) Identify three GPU programming-related mistakes in the CUDA C code in Figure Q4b. Explain and show clearly how they could be fixed.

(6 marks)

Answer

- `__global__` is not used to declare the kernel function as device code
- `__shared__` is not used to declare the temp array in shared memory
- `__syncthreads()` is not used to synchronizes all threads within a block

- Fixed code:
  Line 1: add `__global__`
  Line 2: add `__shared__`
  Line 5: add: `__syncthreads();`

```
Line
1  __global__ void dot_prod(int N, int *a, int *b, int *c){
2    __shared__ int temp [N];
3    int i = threadIdx.x;
4    temp [i] = a[i]*b[i];
5    __syncthreads():
6    // Thread 0 sums the pairwise products
7    if (i == 0) {
8       int sum = 0;
9       for (int j = 0; j < N; j++)
10        sum += temp [j];
11      *c = sum;
12   }
13 }
```

(ii) If N = 256, explain how the kernel should be launched in terms of the number of block(s) and thread(s). Give brief justification for your answer.

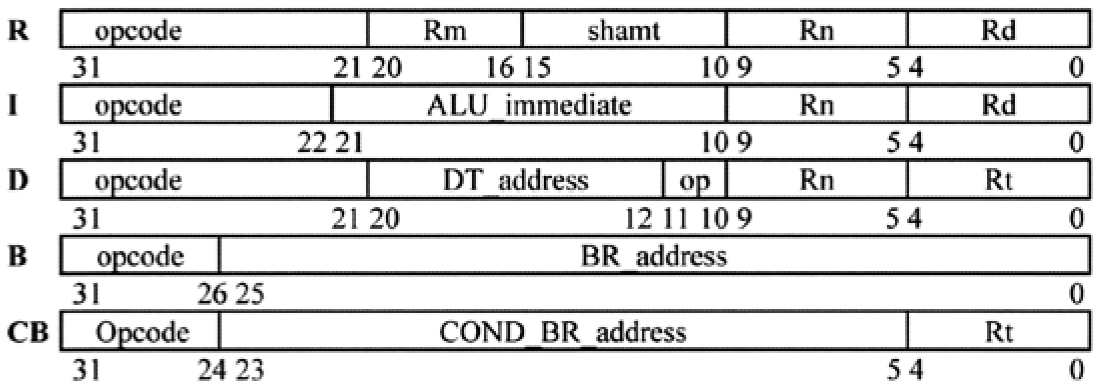
(5 marks)

Answer

- Launch code: dot_prod<<<1, 256>>>(N, d_a, d_b, d_c);

- There are 256 elements so we would need 256 threads
- A thread block can contain up to 1024 threads
- Launching only one thread block avoids the need for additional synchronization between blocks
- No. of warps = 256 / 32 = 8, hence it is aligned with GPU warp sizes

Appendix – Instruction Formats



R	opcode	Rm	shamt	Rn	Rd
	31 21	20 16	15 10	9 5	4 0
I	opcode	ALU_immediate		Rn	Rd
	31 22	21 10		9 5	4 0
D	opcode	DT_address	op	Rn	Rt
	31 21	20 12	11 10	9 5	4 0
B	opcode	BR_address			
	31 26	25 0			
CB	Opcode	COND_BR_address			Rt
	31 24	23 5			4 0