IIT Guwahati - Department of Computer Science & Engineering
CS 223 – Computer Architecture & Organization
Quiz#2 - (12.04.2022): 90 minutes, 30 marks

Solutions

Section I: 4 questions, 3 marks each [4x3 = 12 marks]

1. Given a non-pipelined architecture running at 1.5 GHz, that takes 5 cycles to finish an instruction. You want to make it pipelined with 5 stages. Due to hardware overhead, the pipelined design operates only at 1 GHz. 5% of memory instructions cause a stall of 50 cycles, 30% of branch instructions cause a stall of 4 cycles and LOAD-ALU instruction combinations cause a stall of 1 cycle. Assume that in a given program, there are 20% of branch instructions and 30% of memory instructions. 10% of instructions are LOAD-ALU combinations. What is the speedup of pipelined design over the non-pipelined design if there are no stalls for a non-pipelined architecture (rounded off your answer to two decimal places)?

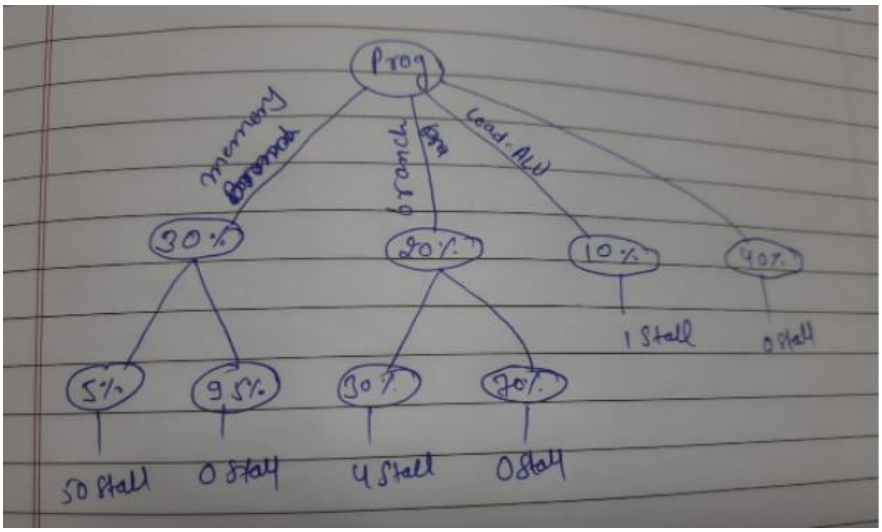
   Non-pipelined:

   CPI-non pipelined = 5 **[1 mark]**

   1 CC – non pipelined= 1/1.5GHz = 0.67 ns.

   Execution Time of 1 instruction = 5 * 0.67 = 3.33 ns

   Pipelined:
   1 CC – pipelined = 1/1GHz = 1 ns



   Base CPI – pipelined = 1

   CPI – pipelined = Base CPI + Stall CPI

   = Base CPI + Memory stalls + branch stalls + LOAD-ALU stalls

   = 1 + [0.3*0.05*50] + [0.2*0.3*4] + [0.1*1]

   = 1+ 0.75 + 0.24 + 0.1 = 2.09 **[1 mark]**

   Execution Time of 1 instruction = 2.09 * 1 = 2.09 ns

   Speed up = 3.33/2.09 = 1.593 ~1.59 **[1 mark]**

2. Assume that the original machine (OM) is an 8-stage pipeline with a 1ns clock cycle. The second machine (SM) is a 12-stage pipeline with a 0.9 ns clock cycle. The 8-stage pipeline experiences 2 stalls due to a data hazard for every 7 instructions, whereas the 12-stage pipeline experiences 3 stalls for every 11 instructions. In addition, branchmis-prediction rate for both machines is 10%. Branches constitute 30% of the instructions.The branch mis-prediction penalty for OM is 2 cycles but for SM it is 5 cycles. What is the speedup of the SM pipeline over the OM pipeline (round off to 2 decimal places)?

8 stage pipeline:

Branch instruction: 30%

CPI considering data hazard = 9/7

No. of stalls due to branch instruction:

*Percentage of branch instructions * Branch miss prediction rate * Branch penalty*

=> 0.3 * 0.1 * 2

=> 0.06

Total stalls per instruction: 9/7 + 0.06 = 1.3457

Avg. execution time:1.3457 * 1

= 1.3457 ns **[1 mark]**

12 stage pipeline:

Branch instruction: 30%

CPI considering data hazard = 14/11

No. of stalls due to branch instruction:

*Percentage of branch instructions * Branch miss prediction rate * Branch penalty*

=> 0.3 * 0.1 * 5

=> 0.15

Total stalls per instruction: 14/11 + 0.15 = 1.4227

Avg. execution time:1.4227 * 0.9

= 1.2804 ns **[1 mark]**

Speed up of SM over OM: 1.3457/1.2804 = 1.050 **[1 mark]**

3. A 32-bit word processor that uses a 32KB 4-way L1 cache and 256 KB 16-way L2 cache has a physical address space of 16 MB. The block size of L1 and L2 caches are 32 and 64bytes, respectively. An L1 cache miss will take 40ns to return the first word from L2 cache and additional 5 ns for each subsequent word.
   (a) Identify the tag, set-index and byte-offset split up of L1 and L2 caches.
   (b) A program P encountered many L1 cache misses. Analysis shows that missed word's position inside a block is uniformly spread across all words. If hit time is 3 ns and hit rate is 0.9, what is the average miss penalty of P if we use (i) Early restart cache optimization (ii) Critical word first cache optimization.

L1	**L2**
32KB	256KB
4-Way	16-Way
32 Bytes	64 Bytes

16MB = 2^**24** [physical address = 24 bits]

32KB/32 = 1024 Blocks 4-Way ⇒ 1024/4 = 256 sets 256= 2^8 32B = 2^5 byte blocks Now, 2^5 blocks ⇒ **5** offset 2^8 sets ⇒ **8** Indices 24-[5+8] = **11** Tag bits	256KB/64 = 4096 Blocks 16-Way ⇒ 4096/16 = 256 sets 256= 2^8 64B = 2^6 byte blocks Now, 2^6 blocks ⇒ **6** offset 2^8 sets ⇒ **8** Indices 24-[6+8] = **10** Tag bits

(a) L1 ⇒ **[0.5 mark]**

11	8	5

L2 ⇒ **[0.5 mark]**

10	8	6

(32*8 byte block)/(32 bit word) = 8 words in a block

(b) Early Restart ⇒ 40 ns if requested word is first word in the block, 45ns if requested word is second word in the block, and so on.

So average miss penalty = [40+45+50+55+60+65+70+75]/8 = **57.5ns [1 mark]**

Critical word first ⇒ **40ns** only. **[1 mark]**

(calculation steps are mandatory)

4. Consider a 5-stage in-order RISC instruction pipeline (IF-ID-EX-MEM-WB). Consider a program that has a sequence of instructions of the form LOAD, ADD, STORE, SUB that is repeated 100 times in that order. So the program has a total of 400 instructions named as I_1, I_2.. I_400. The ADD has a data dependency on the value accessed from memory by the immediate LOAD before it. The STORE and SUB have a data dependency on the resultant of the ADD instruction immediately preceding them. Every LOAD instruction (except the first instruction) has a data dependency on the resultant of immediate SUB before it. Operand forwarding is permitted to reduce data hazards. Assume that the IF for the first LOAD is done at clock cycle number 101.
   (a) What is I_191 instruction?
   (b) In which clock cycle number the 30th STORE performs MEM stage?
   (c) What is the average CPI (cycles/instruction) for this program?

	101	102	103	104	105	106	107	108	109	110	111	112
Load	IF	ID	EX	MEM	WB							
Add		IF	ID	--	EX	MEM	WB					
Store			IF	IF	ID	EX	MEM	WB				
Sub					IF	ID	EX	MEM	WB			
Load						IF	ID	EX	MEM	WB		
Add							IF	ID	--	EX	MEM	WB

(a) Floor (191/4) = 47.

47x4 = 188 so I-188 is SUB

Hence I_189 is LOAD, I_190 is ADD and I_191 is STORE.

Instruction I_191 is STORE **[1 mark]**

(b) 1st LOAD finishes at 105, 2nd at 110...

So, 30th LOAD finishes at 105 + 29x5 = 250th cycle

And it takes 2 more cycle to perform MEM of STORE.

So, Ans is 250+2 = 252nd cycle. **[1 mark]**

(c) 1st SUB finishes at 109, 2nd at 114...

So, 100th at 109+99x5 = 604

Total cycles taken for execution= 604-100 = 504 cycles

CPI = 504/400 = 1.26 **[1 mark]**

Section II: 3 questions, 6 marks each [3x6 = 18 marks]

5. An embedded processor uses 32-bit words for accessing its 16MB main memory system. A program that performs a matrix scan operation resides completely in set# 0 of a 4KB direct mapped cache memory that has a block size of 16 words. The program does the operation C[i]=Max{A[i],B[i]} with the help of a loop that iterates the value of i from 0 to 63. The largest among A[i] and B[i] for each i gets copied to C[i]. A, B, and C are one dimensional arrays of 64 words each and are assigned with physical memory addresses starting from 0x412888, 0x72A900, and 0x236770, respectively. Data words of A, B and C are brought to the cache upon demand and are referred once per iteration in the order B, A and C. Assume that the cache is initially empty except the set# 0 that holds the program.
   (a) Identify the tag, set index and offset split-up of the physical address.
   (b) Identify the number of data misses encountered during the execution of the program.
   (c) Which all words of A, B and C are residing in the cache after the completion of the matrix operation?

Size of the main memory = 16MB (2^{24}B)

Physical address length = 24 bits

Cache size =4KB

(a) set index bits = Cache size / Block size = 4KB/64B = 6 bits

Byte offset = 6 bits

Tag = 12	Set index = 6 bits	Byte offset = 6 bits (4 bit word and 2 bit byte within word)

[1 mark]

(b) For Array A,

starting address 0x412888 = 0100 0001 0010 **1000 1000 1000**

Set no: **34**

Word number : 2 [first 4 bits of byte offset]

First 14 words of array A will be in set 34, next 16 words will be in set 35 and so on...

For Array B,

starting address 0x72A900= 011100101010**10010000 0000**

Set no: **36**

Word number : 0 [first 4 bits of byte offset]

First 16 words of array B will be in set 36, next 16 words will be in set 37 and so on...

For Array C,

starting address 0x236770 = 0010 00110110 **011101100000**

Set no: **29**

Word number : 12 [first 4 bits of byte offset]

First 4 words of array C will be in set 29, next 16 words will be in set 30 and so on...

Array elements	#Set	Array elements	#Set	Array elements	#Set
A[0]-A[13]	34	B[0]-B[15]	36	C[0]-C[3]	29
A[14]-A[29]	35	B[16]-B[31]	37	C[4]-C[19]	30
A[30]-A[45]	36	B[32]-B[47]	38	C[20]-C[35]	31
A[46]-A[61]	37	B[48]-B[63]	39	C[36]-C[51]	32
A[62]-A[63]	38			C[52]-C[63]	33

There will be data misses during the access of the following words:

A0, A14, A30, A46, A62

B0, B16, B32, B48

C0, C4, C20, C36, C52

Total: 14 misses **[2 Marks]**

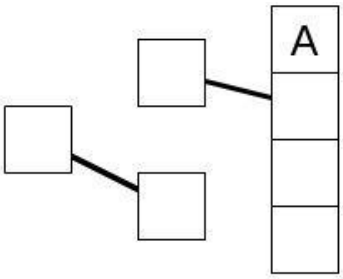
(c) words of A, B and C are residing in the cache after the completion of the matrix operation are

A[0]-A[29]

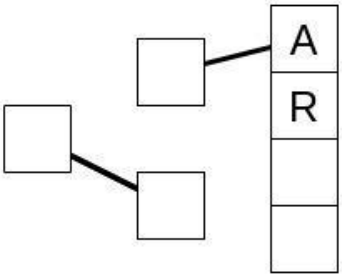
B[0]-B[63]

C[0]-C[63] **[3x1 = 3 Marks]**

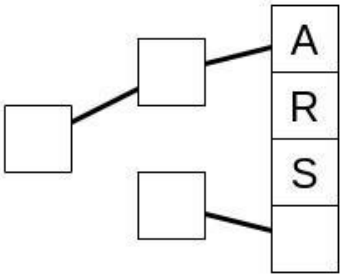
6. Consider a 4-way associative cache that uses pseudo LRU block replacement policy. Assume all the cache blocks are initially empty and filling up of empty blocks in a given cache set happens from way 0 to way-3. Consider the following 13 block numbers (excluding #, $ and @) all mapped to a particular set n given in the order of arrival.
   A, R, S, R, B, A, D, C, S, A, S, B, E, #, $, @
   (a) Find the golden miss ratio of set n, if it is defined as the ratio of compulsory miss to conflict miss in the above sequence of 13 block requests.
   (b) Considering the 13 block requests, what is the hit rate for set n?
   (c) Consider the 14^{th}, 15^{th} and 16^{th} requests #, $ and @, respectively given above for set n. If access to # resulted in a conflict miss, and access to @ resulted in replacement of block 'S', give the set of all possible values that # and $ can have such that request to $ is a hit.



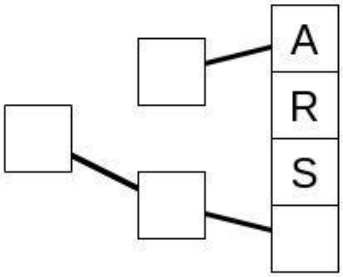
Compulsory Miss = 1
Conflict Miss = 0
Hit = 0



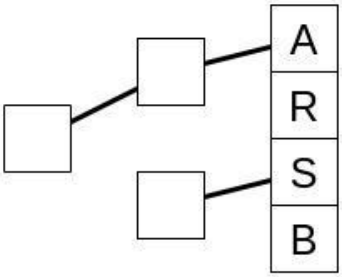
Compulsory Miss = 2
Conflict Miss = 0
Hit = 0



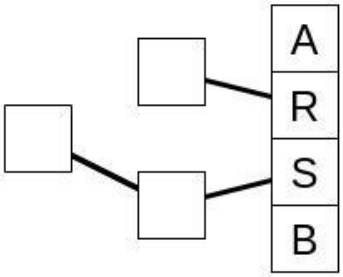
Compulsory Miss = 3
Conflict Miss = 0
Hit = 0



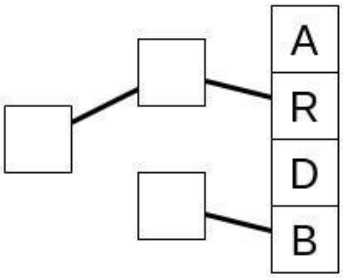
Compulsory Miss = 3
Conflict Miss = 0
Hit = 1



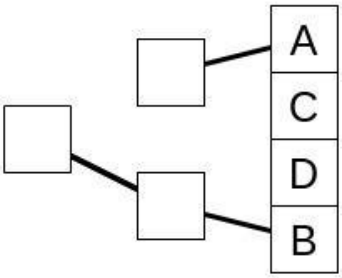
Compulsory Miss = 4
Conflict Miss = 0
Hit = 1



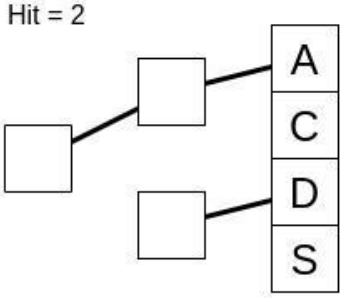
Compulsory Miss = 4
Conflict Miss = 0
Hit = 2



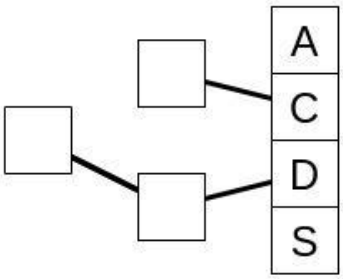
Compulsory Miss = 5
Conflict Miss = 0
Hit = 2



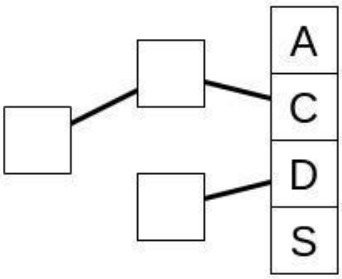
Compulsory Miss = 6
Conflict Miss = 0
Hit = 2



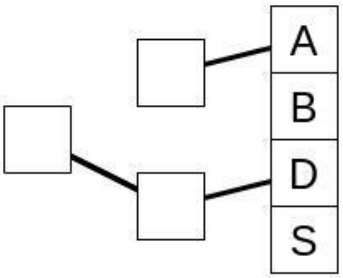
Compulsory Miss = 6
Conflict Miss = 1
Hit = 2



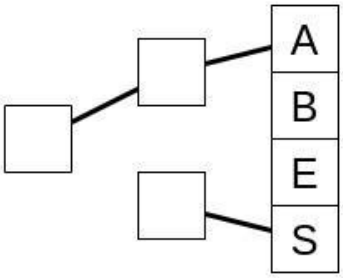
Compulsory Miss = 6
Conflict Miss = 1
Hit = 3



Compulsory Miss = 6
Conflict Miss = 1
Hit = 4



Compulsory Miss = 6
Conflict Miss = 2
Hit = 4



Compulsory Miss = 7
Conflict Miss = 2
Hit = 4

For 4 way set associative cache, three bits are required for PLRU tree.
[Tree illustration is mandatory]

(a) Golden Miss Ratio = Compulsory Miss / Conflict Miss = 7/2 **[2 Marks]**
(b) Hit Rate = 4/13 **[1 Mark]**
(c) Set of all possible values of # and $
can be {R, C, D}.
If $ is a hit, then $ can be either B or anyone of #.
Hence there are 6 combinations for (#,$) = { (R,B), (C,B), (D,B), (R,R), (C,C), (D,D) } **[3 Marks]**

7. In a program there was a branch instruction which is iterated 6 times. The processor uses a (2, 2) branch predictor. The outcome of the last two branches is used to index into the BHT. Each entry of BHT is a 2-bit value that is updated by a standard 2-bit, 4-state finite state machine. Let the initial entry of the BHT for NN/NT/TN/TT is 00/00/11/11, respectively. The BHT is indexed with an NN value initially. The branch predictor has an accuracy of 50% with every alternate iteration of the branch predicted correctly. The first iteration of the branch was mis-predicted. Illustrate the operation of the branch predictor with the help of neat labelled table entries.
   (a) List the actual outcomes of the 6 iterations of this branch instruction in order.
   (b) How many times the predictor gave the output as 'taken'?
   (c) What is the BHT entry after the 6th iteration of this branch instruction?

Sl. No	Last Outcome	BHT NN/NT/TN/TT	Prediction	Outcome	Misprediction (Yes/No)
1	NN(Initially)	00/00/11/11	N	T	Yes [Given]
2	NT	01/00/11/11	N	N	No
3	TN	01/00/11/11	T	N	Yes
4	NN	01/00/10/11	N	N	No
5	NN	00/00/10/11	N	T	Yes
6	NT	00/00/10/11	N	N	No
		00/00/10/11			

[Table illustration is mandatory]

(a) T N N N T N **[2 marks]**
(b) 1 **[2 marks]**
(c) 00/00/10/11 **[2 marks]**