CSCI 540 Homework Assignment 3

Name: Rama Krishna Kamma
CWID: 50321021

----

1. **What do MIR, MPC, MAR, and MDR do in Mic-1 microarchitecture? (10%)**

Answer:

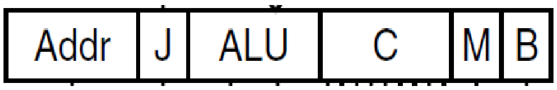
Memory and the machine can exchange information in two ways:
a. A memory port with a word address of 32 bits.
b. a memory port with a byte address of 8 bits.

Two registers, **MAR** and **MDR**, regulate the 32-bit port.

MAR (Memory Address Register): A 32-bit address register is an MAR. Because MAR has word addresses, the numbers 0, 1, 2, etc. refer to a series of words.

MDR (Memory Data Register): A data register is an MDR. The data is read from the address location when a word address is entered into MAR and written into MDR.

MIR (Micro Instruction Register): The purpose of the is to store the current microinstruction.



whose bits activate the data path's control signals.
Addr - The address of the potential following microinstruction is contained.
JAM - Controls how the following microinstruction is chosen.
ALU - ALU and shifter functions.
C - Chooses which registers should be written to via the C bus.
Mem - Memory operations.
B - Chooses the source for the B bus.

MPC (Micro Program Counter): The following program instruction is located in the MPC. When an instruction has completed running, the MPC value is loaded into MAR. MPC will update its value with the address of the upcoming instruction.

2. **What do A Bus, B Bus, and C Bus do in Mic-2 microarchitecture? (5%)**

Answer:

A and B operand buses make up the Mic-2's three-bus design, while bus C serves as an output bus for the results of the Alu driver through a shifter and are then written into registers.

Due to the ability to fetch both operands during a single CPU cycle, the implementation of two operand buses eliminates the requirement for the h register.

3. **Where is the cache physically located? What kinds of localities do cache rely on? Briefly explain the localities. (10%)**

Answer:

Cache speeds up access to memory words by storing them in a compact, quick memory. Utilizing multiple caches is one of the best strategies to increase bandwidth and reduce latency. Adding a separate cache for instructions and data is a simple trick that works quite well. Having separate instruction and data caches, also referred to as a split cache, has a number of advantages.

There are primarily three levels (L-1, L-2, and L-3) of cache:

L1: The quickest cache is L1, which is located on the CPU chip.

L2: It might be located between L1 and main memory in L2. It is not on the CPU, but rather close to it using a high band connection inside the CPU package.

L3: It is faster than DRAM (PRIMARY MEMORY) on the mother board.

Two types of address locality are necessary for caching:

1. Temporary Locality
2. Spatial Locality

Spatial Locality: The understanding that memory locations with equal numerical addresses are likely to be accessed soon if they are close to a memory place that was just accessed.

Temporary Locality: As soon as previously used memory locations are accessed once more, temporary locality happens.

4. **What is a cache line? What is a cache hit? What is a cache miss? What will happen when there is a cache miss? (10%)**

Answer:

Cache lines: It is fixed-size divisions of the main memory.

Typically, 4 to 64 consecutive bytes make up a cache line. When lines are 32 bytes long, line 0 contains line 1 has the bytes 32 to 63, followed by the values 0 to 31. Lines are numbered sequentially beginning at 0.

The 11 LINE bits from the memory address that the CPU generates are required by the hardware in order to index into the cache and locate one of the 2048 entries. To determine whether a cache entry is genuine, the TAG field of the memory address and the Tag field are compared.

Cache Hit the desired phrase can be found in the cache entry.
Cache Miss is employed when the necessary there is no entry in the cache. The cache line of 32 bytes is in this case taken from memory and replaced with the one already present in the cache entry.

5. **If on the same machine, the instructions can have different lengths, what are the advantage(s)? What are the potential problem(s)? (10%)**

Answer:

Yes, we can give the same machine instructions of various lengths. The following are benefits and potential drawbacks of doing this:

Advantages:

- The offered instructions will be more adaptable.
- We can utilize many different registers, memory location, and addressing mode combinations.
- With different combinations of register and memory references as well as addressing modes, addressing can be more versatile.

Potential Issues:

- The complexity of the CPU will rise.
- can impair the processor's ability to operate efficiently.

6. **Convert the following infix notations to postfix notations. (5%)**
(1) A×B+C÷D
(2) A+B+C+D
(3) A+B×(C-D)

Answer:

(1) A×B+C÷D
((A×B)+(C÷D))
(ABx)+(CD÷)
ABxCD÷+

(2) A+B+C+D
(((A+B)+C)+D)
((AB++C)+D)
AB+C++D
AB+C+D+

(3) A+B×(C-D)
A+(Bx(C-D))
A+(Bx(CD-))
A+(BCD-x)
ABCD-x+

7. On a machine, an instruction is always 16 bits long, and there are 16 registers.
(1) Briefly explain the idea of expanding opcode. (5%)

Answer:

In the event that there are (n+k) bit instructions, k bit opcodes, and n bit addresses, 2k distinct operations and 2n addressable memory cells are feasible. Now that the instruction length is constant, we can alter the opcode length by increasing it or decreasing it, i.e., k+1 opcode bits or k-1 opcode bits, whereas the address bits will remain constant at n-1 or n+1, respectively.
By doing this, we reduce the number of addressable cells while increasing the number of instructions required. This method is referred to as opcode expanding.

(2) Design a scheme to support 7 three-address instructions, 31 two-address instructions, 14 one-address instructions, and 32 zero-address instructions. (10%)

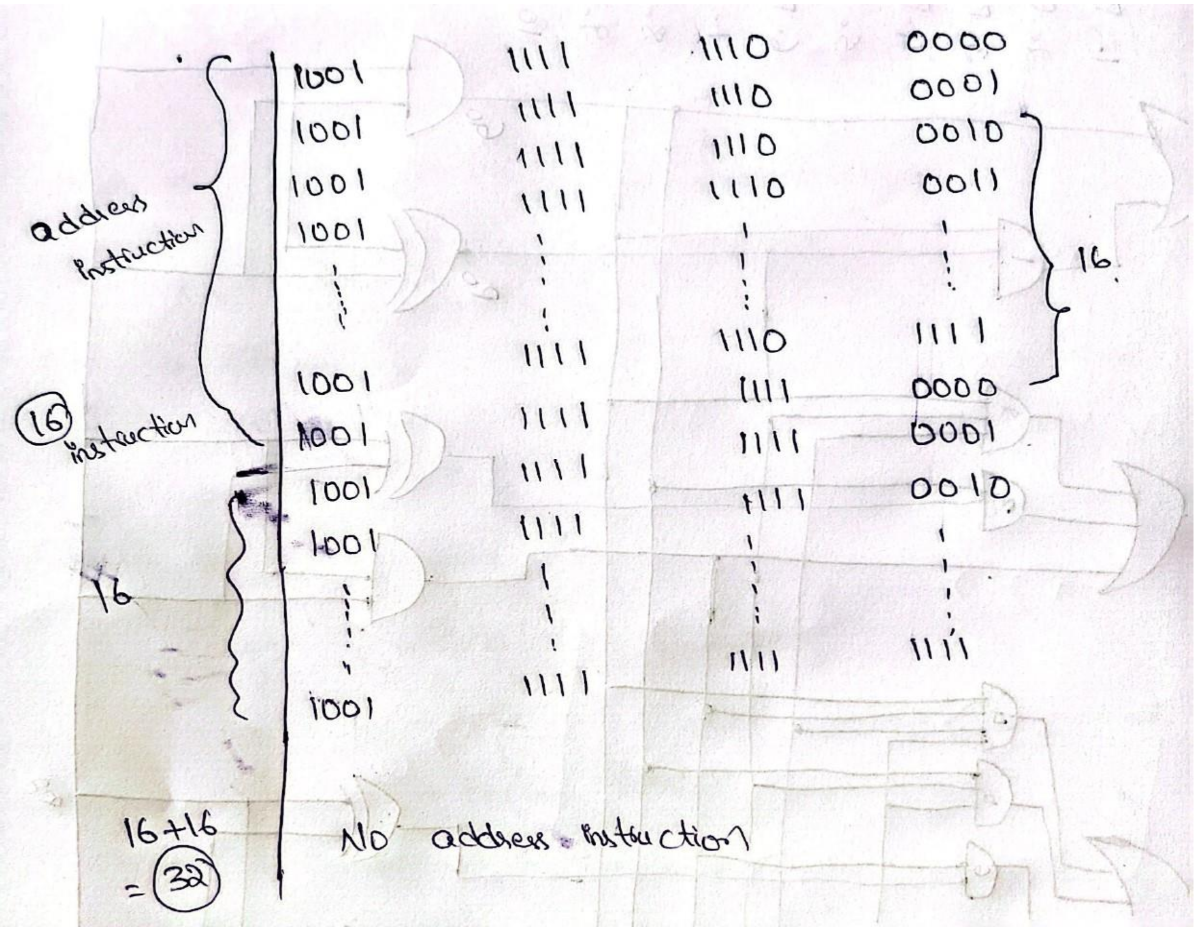
Name : Rama krishna Kamma

CWID : 50321021

Question : 7(2):

4 bit	0 ←	0000	XXXX	YYYY	ZZZZ
opcode	1 ←	0001	XXXX	YYYY	ZZZZ
7		0010	XXXX	YYYY	ZZZZ
3 address		0011	XXXX	YYYY	ZZZZ
instruction		⋮	⋮	⋮	⋮
	7 ←	0110	XXXX	YYYY	ZZZZ
Opcode		0111	0000	YYYY	ZZZZ
16		0111	0001	YYYY	ZZZZ
		0111	0010	YYYY	ZZZZ
		0111	0011	YYYY	ZZZZ
16 + 15 = 31		⋮			
2 Address instructions		0111	0110	YYYY	ZZZZ
		0111	0111	YYYY	ZZZZ
		0111	1000	YYYY	
				YYYY	ZZZZ
		0111	1110	YYYY	ZZZZ
		0111	1111	YYYY	ZZZZ
		1000	0000	YYYY	ZZZZ
15		1000	0001	YYYY	ZZZZ
				YYYY	ZZZZ
		1000	1110	YYYY	ZZZZ
		1001	1110	0000	ZZZZ
		1001	1110	0001	ZZZZ
14 address instructions				1101	ZZZZ
		1001	1110		

Scanned with CamScanner



8. **Assume that the cache size is 256kB, and each cache line is 64 Bytes. (10%)**
   (1) Let us assume this is a Two-way associative cache. How many cache sets are there?

Answer:

According to the given information in the question
Cache size = 256kB = 2^18 bytes
Size of cache line = 64Bytes = 2^6 bytes

Number of cache lines in the cache = Cache Size/size of cache line
= 2^18/2^6
= 2^12 lines.

Two-way associative cache, there will be two lines in each cache set.
Thus, number of cache sets = Number of cache lines in the cache/2
= 2^12/2
= 2^11 sets
= 2048 sets.

Total no of cache lines= 256*2^10 Bytes/2^6 Bytes=2^12
Total no of cache sets= 2^12/2=2^11

(2) Let us assume the following memory blocks need to be accessed:
Memory Blocks #4100, 8196, 2052, 8196, 2052, 4100
If the cache is initially empty, what is the cache hit/miss rate?

Answer:

According to the given question information
Blocks of memory: 4100, 8196, 2052, 8196, 2052, and 4100
Function of mapping: N mod S = i

- The cache is empty, hence 4100 mod 2048 is 4 misses.
- 8196 mod 2048 equals 4 errors.
- Use the LRU technique to substitute 4100 because 2052 mod 2048 is 4 misses.
- 8196 mod 2048 equals 4 hits.
- 2052 mod 2048 equals 4 hits.
- Using the LRU technique, change to 8196 because 4100 mod 2048 is 4 misses.

2 hits total
4 misses in total.

Hit ratio=2/6=33.33%
Miss ratio =4/6=66.66%

9. Given this piece of code fragment: (10%)

```
for (int x = 0; x < 10; x++)
{
    if ( x % 4 >= 2)
        cout << "OK" << endl;
}
```

Assuming the branch prediction by default is "TAKEN",

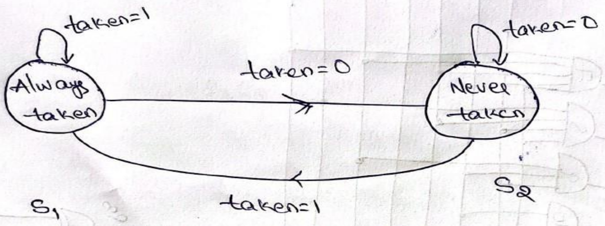
(1) What is the accuracy of branch prediction of the cout statement when we use a 1-bit branch history?

Answer:

Name : Rama krishna Kamma

CWID : 50321021

Question - 9(1):



If (x% 4 >= 2) is present and the values of x range from 0 to 9, then the branch will not be taken since the default prediction is used that it will fail because we are at the taken stage.

The forecast is now "branch not taken."

Since we are currently at, x=1, 1% 4 =1, and it is not greater than 2, the branch won't take not taken stage, thus it will move on.

The prediction will remain unchanged and at the branch not chosen as of right now.

Branch will be taken if x=2, 2% 4 =2, and it is greater than 2, but we are now at not taken stage, hence it will not succeed.

The forecast will now shift to the branch chosen.
Because it is greater than 2 and the branch will take if x=3, 3% 4 = 3, and we are now in the taken stage, it will pass.

The prediction will remain unchanged at the current branch taken.
Since 4% 4 = 0 for x=4 and it is not greater than 2, the branch won't take and we are already in the taken stage, thus it will fail.

The forecast will now read "branch not taken."
Since the branch won't take if x=5, 5% 4 = 1, and it is not greater than 2, and we are now in the not taken stage, it will pass.

The prediction won't change at this time, and the branch will not be taken.
Since we are currently at the not taken stage and the branch will take if x=6, 6% 4 = 2 and it is >= 2 hence the equation will fail.

The forecast will now shift to the branch chosen.

If x=7, then 7% 4 = 3 and it is greater than 2, meaning the branch will take. Since we are already at the taken stage, it will pass.

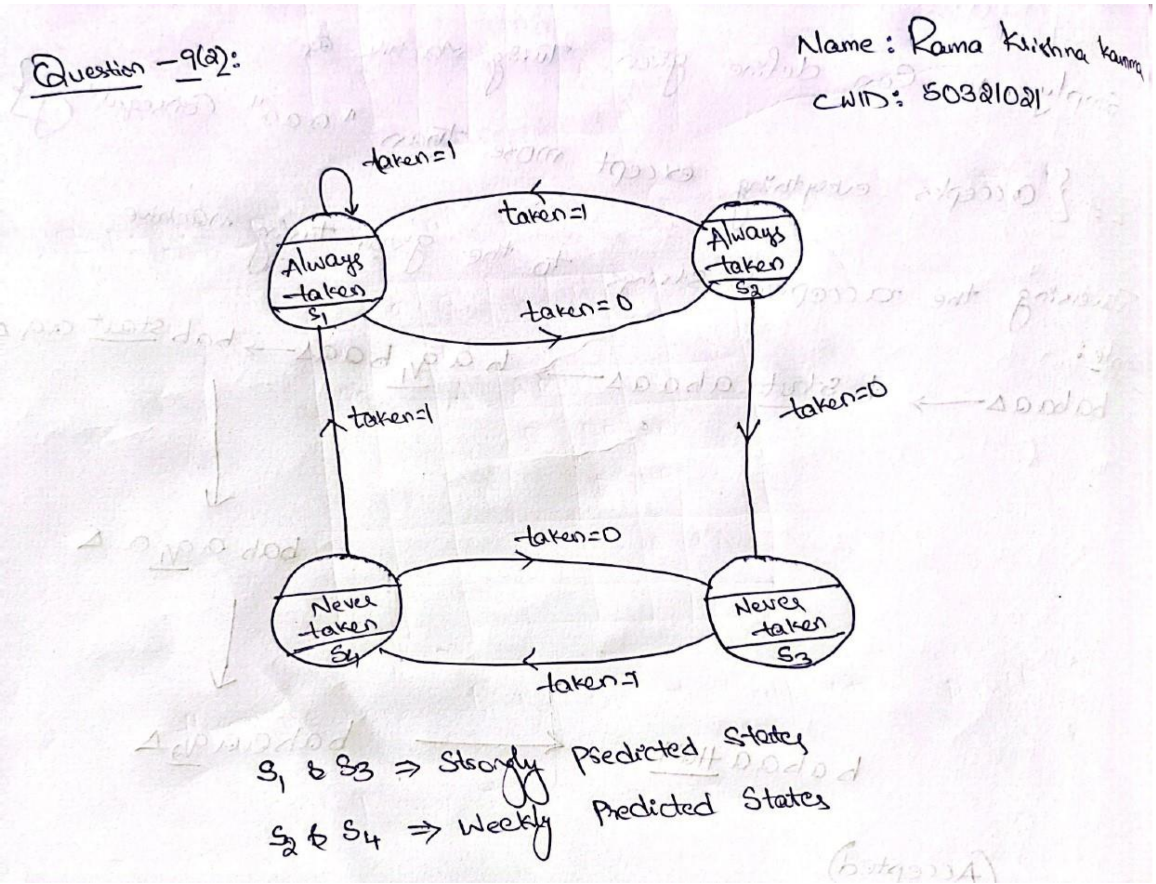
The prediction will remain unchanged at the current branch taken.
Since we are currently in the taken stage and 8% 4 = 0 for x=8 and it is not >= 2, the branch won't take, and the operation will fail.
The prediction will now change, and the branch will not be taken.

Branch won't take if x=9 since 9% 4 = 1 and it is not greater than 2, and we are now at not taken stage, thus it will move on.

The prediction won't change at this time, and the branch will not be taken.
5 out of 10 cases were successful, hence our accuracy is 50%.

(2) What is the accuracy of branch prediction of the cout statement when we use a 2-bit branch history?

Answer:


Question – 9(2):

Name : Rama Krishna kamma
CWID: 50321021

S1 & S3 ⇒ Strongly Predicted States
S2 & S4 ⇒ Weekly Predicted States

Strong predictions include S3 and S1.
Weak predictions exist for S2 and S4.

We have the conditional statement if (x% 4 >= 2) and the values of x range from 0 to 9, therefore the branch won't take since the default prediction is taken, which implies we are at stage S1 and the condition will fail.

Now that we are at taken stage S2, the forecast has changed to branch taken S2 and will fail since for x=1, 1% 4 =1 and it is not >= 2.

Since x=2, 2% 4 =2, and it is greater than 2, the branch will take, however we are already at the not taken S3 stage, the prediction will now change to not taken S3.

Now since x=3 and 3% 4 = 3 and it is >= 2, the prediction will change to branch taken S4. Since we are currently at taken S4 stage, it will fail.

Now that we are at branch taken S1, the prediction will alter and it will fail because, for x=4, 4% 4 = 0, and it is not >= 2, therefore the branch won't take.

Now that we are at the branch taken S2 stage, the prediction will change to branch taken S2. Because x=5, 5% 4 = 1, and it is not >= 2, the branch won't take and it will fail.

As x=6, 6% 4 = 2, and it is >= 2, the branch will take, and since we are currently at the not taken S3 stage, it will fail.

The prediction will now change.
Now that x=7, 7% 4 = 3, and it is >= 2, the branch will take, and since we are at the branch not taken S4 stage at this time, it will fail.

Now that we are at branch taken S1, the prediction will change and fail since x=8, 8% 4 = 0, and it is not >= 2, thus the branch won't take, and we are currently at taken S1.

When x=9, 9% 4 = 1, and it is not >= 2, the prediction will alter, and it will be branch taken S1. Since we are currently at taken S2 stage, it will fail.

The forecast will remain unchanged and will be at branch S3 that was not taken.
Our accuracy is 0% since out of 10 cases, 0 cases were successful.

10. Assume we have 8 registers, R0~R7, and we have a pipeline of 6 stages: Instruction Fetch (IF), Instruction Issue (II), Operands Fetch (OF), Execution (EX), Write Back (WB), and Commitment (CO). Each stage needs exactly 1 cycle to finish its work.

    Also assume that the pipeline supports forwarding, which means the result of WB can be forwarded to OF.

    Given the following piece of instructions:

    ```
    R1 = R0 + R2
    R3 = R1 + R4
    R1 = R5 – R6
    R5 = R0 + R7
    ```

    (1) Identify all the data dependencies and their types. (5%)

Answer:

RAW - read after write is appropriate because we are using R1, which was recently written, in the instructions.
The third instruction thus has R1 writing, so WAR, or write after read, is appropriate.
Given that R1 has had his address changed twice, WAW—write after write—applies.

So, 3 data dependencies possible are,
RAW means read after writing.
WAR stands for write after read.
WAW means Write after write.

- R1 being the first value provided and is then called I2, I1-I2 is now a RAW dependence.
- R5 is called in I3 and given a value in I4, making I3 and I4 a WAR dependency.
- I1-I3 are WAW because R1 received address renaming instructions twice.

(2) How many cycles do we need if we run the instructions in a 6-stage pipeline that does not support forwarding, register renaming, and out-of-order execution? (II stage only decodes the instruction but does not rename its registers.) (5%)

(3) How many cycles do we need if we run the instructions in the 6-stage pipeline that supports forwarding, register renaming, and out-of-order execution? (5%)

Answer for 10(2) and 10(3):

Given Instructions,

I1 = R1 = R0 + R2 I1 : ADD R1,R0,R2
I2 = R3 = R1 + R4 I2 : ADD R3,R1,R4
I3 = R1 = R5 – R6 I3 : SUB R1,R5,R6
I4 = R5 = R0 + R7 I4 : ADD R5,R0,R7

It is necessary to disregard the RAW (read after write) and WAW (write after write) dependencies if we need to execute the instructions that have been issued but do not allow out-of-order execution, forwarding, or register renaming.

pipelined execution from below, Now, if RAW and WAW are ignored, it will take 7 cycles to finish the specified instructions.

We only need to create a table with pipelined execution if we need to execute the instructions that have been provided and allow for out-of-order execution, forwarding, and register renaming.

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9
I1	IF	II	OF	EX	WB	CO			
I2		IF	II	OF	EX	WB	CO		
I3			IF	II	OF	EX	WB	CO	
I4				IF	II	OF	EX	WB	CO

Number of cycles required in the pipelined execution is 9 as shown above.