German University in Cairo - GUC
Faculty of Media Engineering and Technology- MET
Department of Computer Sciences
Assoc. Prof. Milad Ghantous

CSEN702-Microprocessors
Winter 2023

Midterm Exam SOLUTION

Bar Code

Instructions: **Read Carefully Before Proceeding.**

1- Non-programmable calculators are allowed
2- Write your solutions in the space provided
3- The exam consists of **(5) questions**
4- This exam booklet contains **(14) pages** including this page
5- Total time allowed for this exam is **(120) minutes**
6- When you are told that time is up, stop working on the test

Good Luck!

Question	1	2	3	4	5	**Σ**
Possible Marks	10	20	12	17	11	**70**
Final Marks						

Question 1 (10 pts)

Circulate the correct answer. Ambiguous circles will not be considered.

1	The idea of temporal locality helps in reducing compulsory misses.	True	False
2	A higher associative cache is said to have a higher hit time. But employing LRU approach can help in reducing that.	True	False
3	Global miss rate is always less than or equal to local miss rate.	True	False
4	A "hit under miss" approach reduces the miss rate by forwarding the missed request to L2 cache where it might hit.	True	False
5	Using the "Early restart" optimization, the best case scenario is when the requested word is at the end of the block	True	False
6	Without scheduling, WAR and WAW hazards are not considered dangerous.	True	False
7	Static scheduling is done by the hardware while dynamic one is done by the compiler.	True	False
8	Register pressure happens when you have many instructions with source operands coming from the same register.	True	False
9	A loop with 11 iterations will always have 1 dangling loop iteration, when unrolled.	True	False
10	Leakage increases with the increase of thickness of the transistors.	True	False

Question 2 (20 pts)

A) If you are a cache designer, and you are aware that majority of the codes will need temporal locality rather than spatial one, would you increase the block size as an optimization. Why or why not? (2 pts) [Answer 1 pt, why or why not 1 pt]

Solution: if temporal locality is needed, means that once you bring the data item, it's needed again and not its adjacent ones. So increasing the block size won't benefit us, but decreasing it would as we will fit more data items in the cache and hence temporal locality works perfect then.

B) Consider the following nested loop with the two matrices A and B. A is a `double`-precision floating point matrix while B is byte `char` matrix. The cache uses write-allocate strategy, has a size 320 Bytes, with block size of 64 bytes. (7 pts)

```
for(i=0; i<8; i++)
   for(j=0; j<8; j++)
         A[i][j] = B[j][i];
```

Argue if loop interchange would benefit the miss rate or not and show your work.
3 pts for simulating loop as is
3 pts for simulating with interchange
1 pt for verdict
Solution:
The current loop requests A[0][0] and B[0][0] first. Elements A[0][0] till A[0][7] are brought to the cache since they need 64 bytes = 1 block.
B[0][0] is also brought along with the entire B matrix as it all occupies 64 bytes. (8x8 matrix which each element 1 byte char).
Subsequent accesses to B are all hits. Subsequent accesses to A are also hits (A0,1, A0,2, A0,3,..... A0,7), the miss on A[1][0] once then all hits till A[1][7]... until we hit A[4][0], when the cache is already full and start replacing.

When loop interchange is done, also A[0][0] is requests and B[0][0]. Elements A[0][0] till A[0][7] are brought and the entire B matrix. But next access is A[1][0] which is a miss. A[2][0] is next and is a miss, A[3][0] also miss, A[4][0] is a miss and we start removing A[0][0] row!
So no, we should not interchange.

C) How does the different instruction sets affect the energy consumption of microprocessors? (2 pts)

Solution: Instruction sets might be very reduced and hence the compiler has to use many instructions to achieve a certain task, requiring more cycles, and longer time, which will increase the energy consumption, compared with advanced sets, which the compiler can use to create a shorter code, making the execution faster and needing less energy overall.

D) Consider the following code:

```
for(i=0; i<100; i++) {

 A[i] = B[i]+ C[i];  //1
 C[i] = B[i]*5;      //2
 B[i] = A[i]/5;      //3
 A[i] = C[i];        //4

}

for(i=0; i<100; i++) {
 B[i]=C[i]*2;
}
```

D.1) Indicate the true hazards, the output dependence and the anti-dependence, using the line numbers. (3 pts) [1/2 pt each]

True hazards:

- Between 3 and 1 on A[i]
- Between 4 and 2 on C[i]

Anti-dependences:

- Between 2 and 1 on C[i]
- Between 3 and 2 on B[i]
- Between 3 and 4 on A[i]

Output dependence:

- Between [1] and [4] on A[i]

D.2) Use renaming to reduce these hazards to the minimum. Re-write the new code. (6 pts) 1 pt for each renaming

```
for(i=0; i<100; i++) {

 A1[i] = B[i]+ C[i];   //1
 C1[i] = B[i]*5;       //2
 B1[i] = A1[i]/5;      //3
 A[i] = C1[i];         //4

}

for(i=0; i<100; i++) {
 B1[i]=C1[i]*2;
}
```

Question 3 (12 pts)

Assume a memory hierarchy system with 3-levels cache, namely, L1, L2 and L3.

- Out of 5000 memory requests issued by the processor, 4600 hit in L1.
- The local miss rate of L3 was found to be 3%.
- It was also observed that only 50 requests reached L3.
- The hit time in L1 is 1 clock cycle and doubles every level.
- The penalty of L3 is 100 clock cycles.

A) Compute the local and global miss rates of caches L1 and L2 and the global miss rate of L3. (4 pts) 1 pt for each answer

Solution:

L1:

Local = 400/5000 = 0.08 or 8% = global miss rate.

L2:

Local = 50/400 = 0.125 or 12.5%

Global = 50/5000 = 1% (or = local L1 x local L2 = 0.08 x 0.125 = 1%)

L3:

Local = given 3%

Global = *local* L1 x L2 x L3 = 0.08 x 0.125 x 0.03 = 0.0003 or 0.3%

Alternative way to global:

L3 receives 50 requests, 3% miss to 50 x 0.03 = 1.5 misses. So global 1.5/5000 = 0.3%

B) Compute the average memory access time of this hierarchy. (2 pts)

1.5 for formula plugging right values and 0.5 for correct answer.

Avg mem access time =

hit time L1 + miss rate L1 x (hit time L2 + miss rate L2 x (hit time L3 + miss rate L3 x penalty))

= 1 + 0.08 x (2 + 0.125 x (4 + 0.03 x 100))

= 1 + 0.08 x (2.875) = 1.23 clock cycles.

C) The misses per instruction in L1 was found to be `0.1 (10%)`. Compute the average percentage of memory-reference instructions in our codes, and the misses per instruction for L2 and L3. (4 pts)

Misses per instruction in L1 =0.1= (400 misses) / (total nbr of instructions)

So total number of instructions = 400/0.1 = 4000 instructions.

We know we have 5000 requests coming from 4000 instructions, which means we have on average 5000/4000 = 1.25 memory request per instruction ➔ 25% of our instructions are loads and stores. 2 pts
For L2: misses per instruction = 50 miss /4000 = 0.0125 1pt

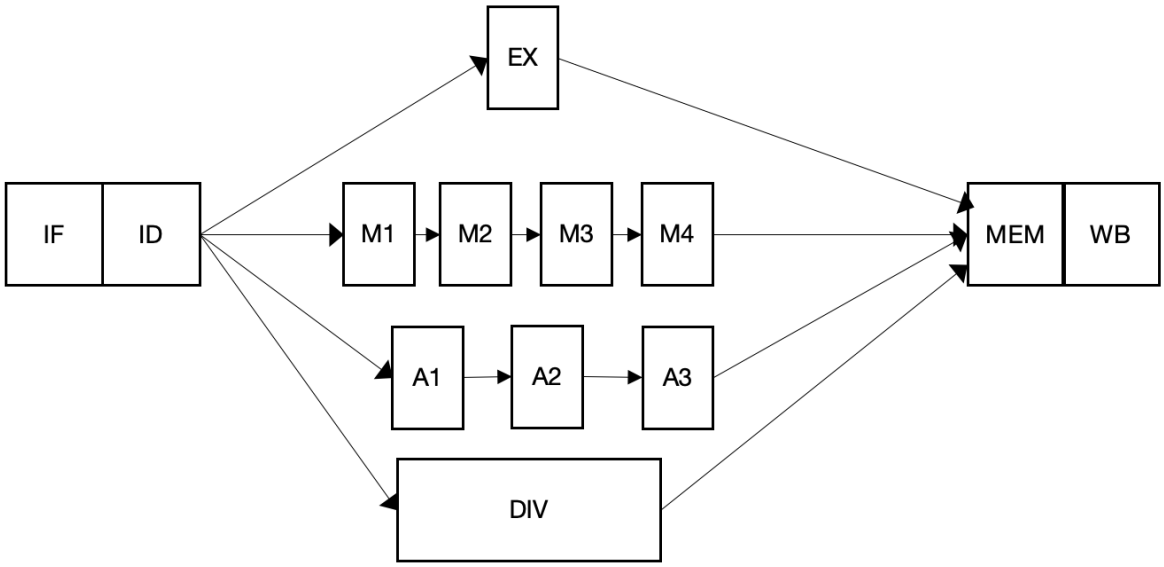
For L3: misses per instruction = 1.5 miss / 4000 = 0.000375 1pt

D) Compute the average memory stalls per instruction in the hierarchy. (2 pts)
1.5 for plugging right values and correct equation and 0.5 for answer.
Misses per instruction L1 x hit time L2 + misses per instruction L2 x hit time L3 + misses per instruction L3 x penalty
= 0.1 x 2 + 0.0125 x 4 + 0.000375 x 100 = 0.2875 cycles.

Question 4 (17 pts)

Given the below modified floating MIPS Datapath where we have the following:
The integer/FP multiplier is pipelined into 4 stages, while the FP adder is pipelined into 3 and the divider is not pipelined but requires 5 cycles to finish.

- Forwarding is applied everywhere.
- Branches are resolved at `decode` stage and they do *not* use a delayed slot.
- Special hardware is available to detect if 2 instructions are safe to use MEM stage or WB stage in the same cycle.



A) Prove that the latencies in the below table are correct, using a timing diagram for each one. (5 pts) 1 each

#	Instruction producing result	Instruction using result	Latency in cycles
A	Load	Any arithmetic	1
B	Multiply	Store	2
C	store	load	0
D	Integer ALU	branch	1
E	Multiply	Any arithmetic	3

Solution:

A)

F	D	X	M	W	
	F	D	STALL	X	

B)

F	D	M1	M2	M3	M4	M	W		
	F	D	X	STALL	STALL	M	W		

C)

F	D	X	M	W	
	F	D	X	M	W

D)

F	D	X	M	W		
	F	STALL	D	X	M	W

E)

F	D	M1	M2	M3	M4	M	W		
	F	D	STALL	STALL	STALL	X			

B) Compute the number of cycles needed for 1 iteration of the following loop, which updates a double-precision array, *without* any scheduling or unrolling. Show your work. F9 and R2 are precomputed. (4 pts) correct stall locations and amounts 3 pts, answer 1 pt

```
LOOP, L.D    F1, 0(R2)
      MUL.D  F2, F1, F9
      S.D    F2, 8(R2)
      DADDUI R2, R2, 8
      BNE    R2, R3, LOOP
```

Solution

```
LOOP, L.D    F1, 0(R2)
      stall
      MUL.D  F2, F1, F9
      Stall stall
      S.D    F2, 8(R2)
      DADDUI R2, R2, 8
      stall
      BNE    R2, R3, LOOP
```

Total = 9 cycles/ iteration

C) Unroll the loop 2 times, without scheduling and compute the speedup gained per itration. (4 pts) correct stalls 3, answer 1

```
L.D F1,0(R2)
STALL
MUL.D F2,F1,F9
STALL STALL
S.D F2, 8(R2)
L.D F3, 8(R2)
STALL
MUL.D F4,F3,F9
STALL
STALL
S.D F4, 16(R2)
DADDUI R2,R2,16
STALL
BNE R2,R3, LOOP
```

Total = 15 for 2 iterations so 7.5 cycles per iteration.

D) Schedule the unrolled loop from part C, to the best of your ability and compute the speedup gained per iteration compared with part C. *Explain* the result. (4 pts)

Correct stalls: 2.5 pts, answer: 0.5 pt, explanations: 1 pt

```
L.D F1,0(R2)
STALL
MUL.D F2,F1,F9
STALL STALL
S.D F2, 8(R2)
L.D F3, 8(R2)
STALL
MUL.D F4,F3,F9
STALL
DADDUI R2,R2,16
S.D F4, 0(R2)
BNE R2,R3, LOOP
```

Total = 13 per 2 iterations so 6.5 per iteration.
LOOP is not parallelized and hence the gains are minimal.

Alternative way: since loop is A[i] = A[i-1]*f9 or (A[i+1] = A[i]*f9)

```
L.D F1,0(R2)
STALL
MUL.D F2,F1,F9
STALL STALL
S.D F2, 8(R2)
MUL.D F3,F2,F9   #remove the second load and use the multiply result directly
STALL
DADDUI R2,R2,16
S.D F3, 0(R2)
BNE R2,R3, LOOP
```

Total = 11 per 2 iterations so 5.5 per iteration.

Question 5 (11 pts)

Consider the following 2-way set associative cache that has a total size of 256 bytes, a block size of 32 bytes. The memory uses 12 bit addresses.

- Show the content of the cache after each of the following instructions is executed.
- **NOTE THAT PARTS ARE DEPENDENT ON EACH OTHER (WHEN YOU SOLVE PART (2), CONSIDER THAT PART (1) HAS TOOK PLACE. AND SO ON)**
- The cache starts empty, and the default way when set is empty, is way 0.
- Register R1 contains value 12

0.5 each entry

1) L.D F2, 20(R1)

Solution: 20 + R1 = 32 = 00000 01 00000 so index is 01, it goes to way 0 by default now. Dirty is 0 since reading, valid is 1 since the block is valid and block replacement is now way 1 since it's the LRU and way prediction is way 0 since it is the MRU.

Set	Block replacement bit	Way prediction bit	Way 0			Way 1		
			Valid bit	Dirty bit	Tag	Valid bit	Dirty bit	Tag
00								
01	1	0	1	0	00000			
10								
11								

2) S.D F3, 149(R1)

149+12 = 161 = 00001 01 00001 so index is also 01, we place it in way 1 since it's the LRU, update LRU to become way 0, dirty bit is 1 since it's writing, and valid is 1.

Set	Block replacement bit	Way prediction bit	Way 0			Way 1		
			Valid bit	Dirty bit	Tag	Valid bit	Dirty bit	Tag
00								
01	0	1	1	0	00000	1	1	00001
10								
11								

3) L.D F4, 404(R1)

404+12 = 416 = 00011 01 00000 so index is also 01, it must go to way 0 since it's the LRU and will replace it. Valid is 1, dirty is 0. Update LRU to way 1, MRU to way 0

Set	Block replacement bit	Way prediction bit	Way 0			Way 1		
			Valid bit	Dirty bit	Tag	Valid bit	Dirty bit	Tag
00								
01	1	0	1	0	00011	1	1	00001
10								
11								

Formula Sheet

Pipelining
$\text{CPI pipelined} = \text{Ideal CPI} + \text{Pipeline stall clock cycles per instruction}$
$\text{Speedup} = \dfrac{\text{Pipeline depth}}{1 + \text{Pipeline stall cycles per instruction}}$
Power and Performance
$\text{Energy}_{\text{workload}} = \text{average power x execution time for the workload}$
$\text{Energy}_{\text{dynamic}} \propto 1/2 \times \text{Capacitive load} \times \text{Voltage}^2$
$\text{Power}_{\text{dynamic}} \propto 1/2 \times \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency switched}$
$\text{Power}_{\text{static}} \propto \text{Current}_{\text{static}} \times \text{Voltage}$
$\text{Geometric mean} = \sqrt[n]{\prod_{i=1}^{n} sample_i}$
Memory hierarchy
$\text{CPU execution time} = (\text{CPU clock cycles} + \text{Memory stall cycles}) \text{ x Clock cycle time}$
$\begin{aligned}\text{Memory stall cycles} &= \text{Number of misses} \times \text{Miss penalty} \\ &= \text{IC} \times \frac{\text{Misses}}{\text{Instruction}} \times \text{Miss penalty} \\ &= \text{IC} \times \frac{\text{Memory accesses}}{\text{Instruction}} \times \text{Miss rate} \times \text{Miss penalty}\end{aligned}$
$\dfrac{\text{Misses}}{\text{Instruction}} = \dfrac{\text{Miss rate} \times \text{Memory accesses}}{\text{Instruction count}} = \text{Miss rate} \times \dfrac{\text{Memory accesses}}{\text{Instruction}}$
$\text{Average memory access time} = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$
$\begin{aligned}\text{Average memory access time} &= \text{Hit time}_{L1} + \text{Miss rate}_{L1} \\ &\times (\text{Hit time}_{L2} + \text{Miss rate}_{L2} \times \text{Miss penalty}_{L2})\end{aligned}$

SCRATCH SHEET