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

CSEN702-Microprocessors
Winter 2022

Midterm Exam

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 **(13) 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	16	14	17	20	**77**
Final Marks						

Question 1 (10 pts)

Mark each of the following statement with T or F.
Ambiguous letters will not be considered ☺

#	Statement	T/F
1	"Early Restart" and "Critical word first" help in reducing miss rate.	
2	Having multi-level caches will help in decreasing the miss rate of L1 cache.	
3	"Way prediction" optimization in set associative caches targets the hit time.	
4	A program with 25 loads and 25 stores, has a 1.5 memory reference per instruction, on average.	
5	A CPU with higher power consumption means it has a higher energy consumption as well.	
6	Turning off the clock for some unused components may help in reducing leakage (static power).	
7	Per second, memory requests and memory accesses are equal.	
8	When `A[0]` is requested, `A` is a single-precision float array, `A[1]` up to `A[7]` were also brought to the cache. The cache block size is 64 bytes.	
9	A write-thru cache is preferred in multi-core systems because data will be always coherent between caches and main memory.	
10	When a CPU issues a read request for a word from a block, but that block's "dirty bit" was found to be 1 in the cache, it's a miss and the clean block needs to be requested from the main memory.	

Question 2 (16 pts)

Answer each of the following parts.

Part 3.1) Assume you had the choice to choose a clock rate for each piece of code you want to execute on your processor. For a specific low-priority code, you decided to use a slower rate to reduce energy and heat. *Will this work? Explain. (4 pts)*

Part 3.2) Consider this loop. *Argue if it can be parallelized or not. Explain your answer.* (4 pts)

```
for(i=1; i<N; i+=2)
{
  a[i] = a[i] + a[i-1]
}
```

Part 3.3) Consider a pipelined CPU that incurs 1 stall every 12 instructions due to data hazards. It employs an always taken prediction scheme for its branches. A Structural stall occurs 3 times every 100 instructions. (4 pts)
Given that the programs contain on average 20% branches that are executed in the decode stage and these branches were found to be taken 90% of the time, *calculate the CPI.*

Part 3.4) Consider a processor that executes codes having 20% floating point instructions, 50% ALU and the remaining are memory instructions.
A Floating point instruction takes 8 cycles on average to complete, while any ALU instruction takes 6 cycles on average, and a memory instruction takes 7 cycles.
We have the following measurements:

- The integer multiply instructions constitute 10% of all ALU instructions and each one requires, alone, 9 cycles.
- The floating multiply instructions constitute 50% of all floating point instructions, and each one, requires, alone, 14 cycles.

What speedup would we gain, against the current CPI, if we were able to decrease the CPI of the integer multiply to 5 cycles and, at the same time, the floating point multiply down to 11 cycles? (4 pts)

Question 3 (14 pts)

Part A) Consider the following code. A is a double-precision floating point array.

- Assume the cache is very large, starts empty, and a block size of 16 bytes.
- The array first element is stored at address 0 in memory.

```
for(i=0;i<=10;i++)
   { A[i] = A[10-i]+1; }
```

1) Consider a cache that uses the "no-write-allocate" strategy. How many *read* and *write misses* will this code incur. Show your work. (5 pts)

2) Compute the number of read *misses and number of write misses* if the system uses "write-allocate" strategy. Also start from cache empty initially. Show your work. (5 pts)

3) Consider that the "no-write-allocate" strategy in part 1, uses a write-buffer of size 16 bytes, and gets transferred to main memory only when full. The buffer is not checked on read misses. Will this pose any data inconsistency issues in the above code or not? Explain. (4 pts)

Question 4 (17 pts)

Consider the following measurements for several cache designs:

I. Unified 64 KB cache: 40 misses per 1000 instructions.
II. 48 KB data cache: 30 misses per 1000 instructions.
III. 32 KB data cache: 25 misses per 1000 instructions.
IV. 16 KB instruction cache: 5 misses per 1000 instructions.
V. 32 KB instruction cache: 8 misses per 1000 instructions.

You have the option to choose one of the three following configurations for a processor:

- Configuration 1: The 64 KB unified cache.
- Configuration 2: The 48 KB data cache with 16 KB instruction cache.
- Configuration 3: The 32 KB data cache with 32 KB instruction cache.

- The hit time in all caches is 1 clock cycle.
- The miss penalty for all caches with sizes up to and including 16KB, is 40 clock cycles, while the penalty for all larger sizes is 120 clock cycles.
- The unified cache is multi-ported and can serve an instruction and data hits in the same cycle.
- 30% of the instructions are loads and stores.

A) Compute the individual miss rates of all caches I through V. (5 pts)

B) Compute the overall miss rate of configuration 2 and the overall miss rate of configuration 3. (4 pts)

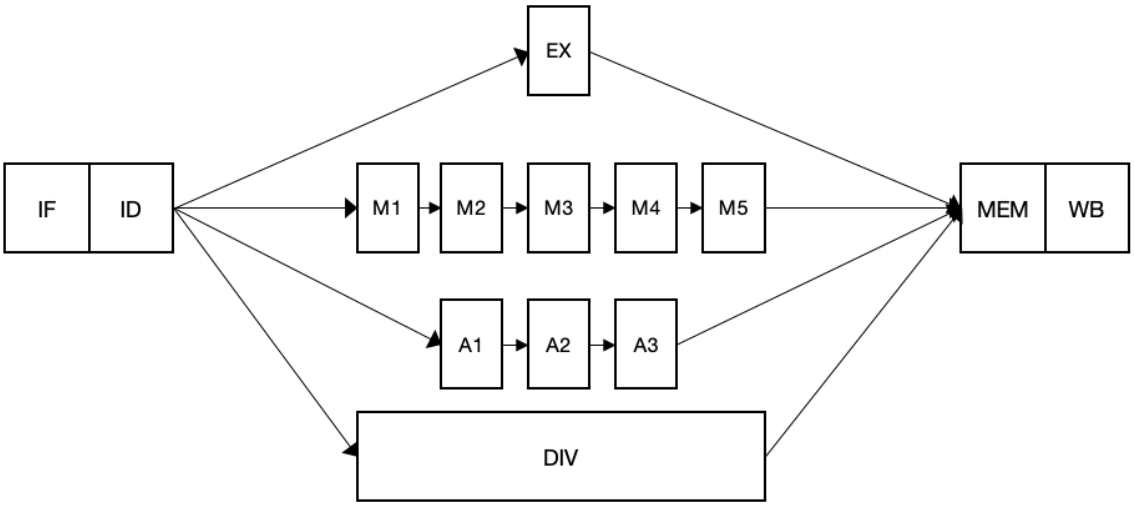
C) Compare the results of all configuration miss rates and discuss your choice. (2 pts)

D) Compute the average memory access time of all configurations and discuss the result. (6 pts)

Question 5 (20 pts)

Consider the following MIPS architecture where the floating point/integer multiplier is pipelined into 5 stages, the floating point adder is pipelined into 3 stages, and the divider is unpipelined and requires 10 cycles. The integer ALU (EX stage) takes 1 cycle and can handle integer arithmetic such as additions and subtractions, address calculations, as well as branch outcome executions.

- Assume forwarding is active everywhere.
- Assume we have a hardware that detects if the data cache is used in the MEM stage or not, by an instruction, and if not, another instruction can use the cache in the same cycle. Also assume the same thing for register file during the WB stage.



A) Compute the number of stalls required between the following. (6 pts)

Mark your answers in the below table.

	Result Producer	**Same result Consumer**	**# of Stalls**
1	Load	Any arithmetic instruction	
2	Multiply	Any arithmetic instruction	
3	FP addition	Store	
4	Integer arithmetic	Branch	

Show your work here

B) Using the latencies you found in part (A), compute the number of cycles needed per 1 iteration of the following code. *Show your work. (4 pts)*

```
LOP,    L.S     F1,  0(R1)
        MUL.S   F2,  F1,  F30
        ADD.S   F3,  F2 , F2
        S.S     F3,  0(R1)
        DADDUI  R1,  R1, -4
        BNE     R1,  R2, LOP
```

C) Unroll 2 times and schedule the loop to the best of your ability, and compute the speedup you gained. (Assume branches do not use delayed slots) (10 pts)

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
$\text{CPU execution time} = (\text{CPU clock cycles} + \text{Memory stall cycles}) \text{ x Clock cycle time}$
$\begin{aligned}\text{Memory stall cycles} &= \text{Number of misses} \times \text{Miss penalty} \\ &= \text{IC} \times \frac{\text{Misses}}{\text{Instruction}} \times \text{Miss penalty} \\ &= \text{IC} \times \frac{\text{Memory accesses}}{\text{Instruction}} \times \text{Miss rate} \times \text{Miss penalty}\end{aligned}$
$\dfrac{\text{Misses}}{\text{Instruction}} = \dfrac{\text{Miss rate} \times \text{Memory accesses}}{\text{Instruction count}} = \text{Miss rate} \times \dfrac{\text{Memory accesses}}{\text{Instruction}}$
$\text{Average memory access time} = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$
$\begin{aligned}\text{Average memory access time} &= \text{Hit time}_{L1} + \text{Miss rate}_{L1} \\ &\times (\text{Hit time}_{L2} + \text{Miss rate}_{L2} \times \text{Miss penalty}_{L2})\end{aligned}$