CSC 230 - Assignment 4 - Part 1 (Written)

Due Friday Aug. 1 by 15:00 in the course box in ECS, but will be accepted (without penalty) until Tuesday Aug. 5 at 13:00

Total Marks = 50

Question 1. [12] In order to compute the hit ratio and effective access time for a multilevel cache, the hits and misses must be recorded among all caches. You are given:

M = the total number of memory accesses;

TM = Main memory access time;

A1 = number of memory accesses found in L1 cache;

T1 = cache L1 hit time;

A2 = number of memory accesses found in L2 cache (i.e. not in L1 and not in Main memory);

T2 = cache L2 hit time;

(a) [2] State an equation for the hit ratio for the L1 cache, H1, relative to M.

(b) [2] State an equation for the hit ratio for the L2 cache, H2, relative to the number of times an access was not found in L1 and is not in Main memory.

(c) [2] State an equation for Tot1, the total time to access L1.

(d) [2] State an equation for Tot2, the total time to access L2.

(e) [2] State an equation for the complete TEFF = Effective Time, using all of the above, i.e. including access to L1, l2 and Main memory.

(f) [2] Show an equation to be used to calculate the possible speedup gained by adding the L1 cache alone.

Question 2. [6] A benchmark program is run on a 40 MHz[1] processor. The executed program consists of 100,000 instruction executions, with the following instruction mix and clock cycle count:

Instruction type	Instruction count	Cycles per instruction
Integer arithmetic	45,000	1
Data transfer	32,000	2
Floating point	15,000	2
Control transfer	8,000	2

(a) [2] Determine the effective CPI (show your calculations).

(b) [2] Compute the execution time (show your calculations).

(c) [2] A common measure of performance for a processor is the rate at which instructions are executed, expressed as millions of instructions per second (MIPS), referred to as the MIPS rate. We can express the MIPS rate in terms of the clock rate and CPI as follows:

$$\text{MIPS rate} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

State the MIPS rate for the benchmark program above (at least show the complete equation even if you cannot handle the computation)

1. 1 second = 10^9 ns and Hz is cycles/sec. MHz = cycles/sec x 10^6

Question 3. [18] A full example of how the cache works was given in some lecture notes to you, showing all the steps from a textbook problem. Here it is reproduced for you.

A computer uses a small direct-mapped cache between the main memory and the processor. The cache has four 16-bit words, and each word has an associated 13-bit tag, as shown in the figure on the right, where, however, the tag has been ignored for this task. When a miss occurs during a read operation, the requested word is read from the main memory and sent to the processor. At the same time, it is copied into the cache, and its block number is stored in the associated tag.

	16 bit content
address 0	
address 2	
address 4	
address 6	

Consider the following loop in a program where all instructions and operands are 16 bits long and the code starts at address 0x02EC:

```
LOOP:ADD   (R1)+,R0        02EC
     DECR R2               02EE
     BNE   LOOP            02F0
```

Assume that, before this loop is entered, registers R0, R1 and R2 contain: R0 = 0, R1 = 0x054E, R2 = 3. Also assume that main memory contains the data as shown on the right

memory address	**16 bit content**
054E	A03C
0550	05D9
0552	10D7

The example went through the execution of the code segment showing how the cache is used in every step. First of all, it makes it easy to decide beforehand the location in cache to which each word is mapped. For the 3 instructions, the calculation was as follows:

```
instruction              address of instruction
LOOP:ADD  (R1)+,R0       02EC =0000 0010 1110 1100     mapped to cache 4
          DECR R2        02EE =0000 0010 1110 1110     mapped to cache 6
          BNE  LOOP      02F0 =0000 0010 1111 0000     mapped to cache 0
```

For the 3 memory locations, the calculation was as follows:

```
content   address in memory
A03C      054E = 0000 0101 0100 1110    mapped to cache 6
05D9      0550 = 0000 0101 0101 0000    mapped to cache 0
10D7      0552 = 0000 0101 0101 0010    mapped to cache 2
```

Looking at the execution, after this first iteration there have been 4 memory accesses and no cache hits. The second iteration is similar, but some memory accesses are avoided, and in total there are 2 memory accesses and 2 cache accesses. At the end, the table below shows the totals for memory and cache accesses. The example also assumes that the access time to main memory is *10t* and that of cache is *1t*. The execution time for each pass, ignoring the time taken by the processor between memory cycles, is calculated in the right column.

(a) [16] Your task now is to repeat the process using separate instruction and data caches, each of the same size as the previous one. You do not need to show every step with explanation as above, only:

1st iteration	4 memory accesses	(10t x 4) = 40t	Total execution time: 75t
2nd iteration	2 memory accesses, 2 cache accesses	(2t x 2) + (2t x 2) = 22t	
3rd iteration	1 memory access, 3 cache accesses	(10t x 1) + (1t x 3) = 13t	

- A diagram of the cache after each iteration is completed.
- A summary of the memory and cache accesses after each iteration is completed.
- A final table showing the total speed of execution.
- A final comment on whether separate caches appear to speed up execution.

Use the tables below to fill in your answers.

After 1st iteration

INSTR. CACHE	
cache line	**16 bit content**
0	
2	
4	
6	

DATA CACHE	
cache line	**16 bit content**
0	
2	
4	
6	

Memory accesses =

Cache accesses =

After 2nd iteration

INSTR. CACHE	
cache line	**16 bit content**
0	
2	
4	
6	

DATA CACHE	
cache line	**16 bit content**
0	
2	
4	
6	

Memory accesses =

Cache accesses =

After 3rd iteration

INSTR. CACHE	
cache line	**16 bit content**
0	
2	
4	
6	

DATA CACHE	
cache line	**16 bi conte**
0	
2	
4	
6	

Memory accesses =

Cache accesses =

Iterations	Total accesses	Total time per iteration	
1st iteration			Total execution time:
2nd iteration			
3rd iteration			

(b) [2] A second possibility being explored is to assume that the cache is used only to store instructions. Data operands are fetched directly from the main memory and not copied into the cache. It appears that one gains faster execution. Can you verify such a claim and give some reason as to why does this choice lead to faster execution than when both instructions and data are written into the *same* cache?

Question 4. [14] Consider a computer system whose main memory (RAM) is byte addressable and 32KB in size. This computer's address bus is 16 bits wide, and uses virtual memory with 4KB pages.

(a) [2] If the processor generates a virtual address of `0x3A72`, what are the page number and the offset components of that address? (Give both answers in hexadecimal.)

Page #: _ _ _ _ _ _ _ _ _ _ _ _ **Offset:** _ _ _ _ _ _ _ _ _ _ _ _

(b) [11] Suppose that a program has been running for a while. A snapshot at an instant in time shows the following pages in main memory (as shown on the right-hand side). Complete the entries in the page table (the table on the left) which would be used by the MMU to map the virtual addresses to real addresses.

Page Number	Valid	Frame Number
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		

page 11	*frame 0*
---	*frame 1*
page 4	*frame 2*
page 0	*frame 3*
---	*frame 4*
page 1	*frame 5*
page 8	*frame 6*
page 5	*frame 7*

Write your answer in the boxes in this table. If the entry for Valid is zero, leave the Frame Number field blank.

(c) [1] A TLB (*Translation Lookaside Buffer*) is normally used to accelerate the translations from virtual addresses to real addresses. Assuming now 16 bit addresses and pages only 2KB in size, suppose the TLB contains the entries shown below. To what real address is the virtual address translated?

Page Number	Valid	Frame Number
7	1	3
3	1	1
6	1	7

Virtual Address = 313C

Real Address = _ _ _ _ _ _ _ _ _ _ _ _