

Test 2 (100 pts)

Grading policy:

$\frac{1}{4}$ Credit for correct answer only

$\frac{3}{4}$ Credit for well written and solid justification/facts/arguments. **Show your work.**

For Test 2, we assume a CPU with a 32-bit address bus.

A) Cache

- 1) **(8 points)** We consider a 2-way associative 32 KB cache with 4 KB blocks. The CPU generates the address 0x204865.
 - a. **(3 points)** What is the block address (in Hexa **OR** binary) of Block **B** containing address 0x204865?
 - b. **(3 points)** What is Block B's tag (in Hexa **OR** binary)?
 - c. **(2 points)** What is the index where Block **B** will be stored in the cache?
- 2) **(12 points)** Consider the cache on the figure below. Answer the following questions based on the figure below

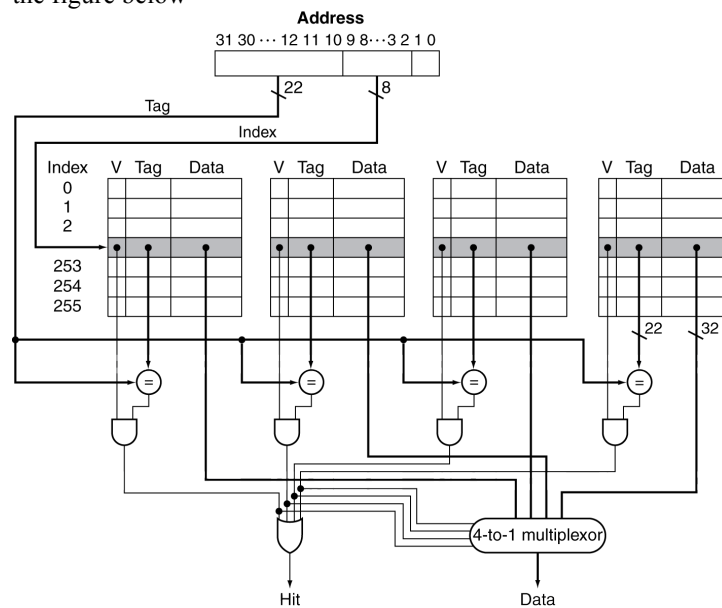


Figure 1

- a. **(2 points)** Is the above cache 2-way associative cache? Explain why

- b. **(10 points)** Redraw Figure 1 (above) such that the cache becomes 2-way associative with a block size of 64 bytes and a cache size of 64 KB of data. Just like for Figure 1, **indicate clearly the size of the busses as well as the smallest and largest Indexes.**

Exercise 1: (30 points) We assume that the data cache size is 2 KB (data) and the block size is 32 bytes. We assume that $M[i][j]$ is adjacent to $M[i][j+1]$ in the memory and that the cache is fully associative using LRU replacing. Consider the following code:

```
char A[64][64];
int i, j;
for (j = 0; j < 32; j++) {
    for (i = 0; i < 64; i++)
        A[i][j] = A[i][j] ^ A[i][n-1-j];
        A[i][n-1-j] = A[i][j] ^ A[i][n-1-j];
        A[i][j] = A[i][j] ^ A[i][n-1-j];
}
```

This code performs a SPECIAL column-wise transpose. As indicated on **Figure 2**, the transposition is performed as follows: the first column ($j=0$) and the last column ($j=63$) are swapped, the second column ($j=1$) and the column ($j=62$) are swapped, and so on...

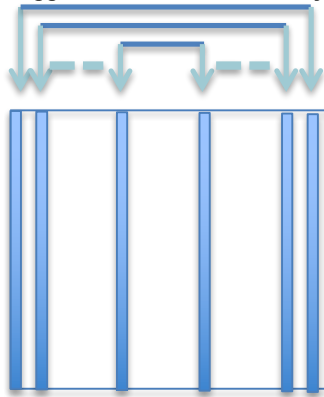


Figure 2

We assume that the code is in a separate instruction cache and the variables i and j are in registers.

- a) **(2 points)** How many blocks are needed to store one line of the matrix?

- b) **(2 points)** How many blocks are needed to store the full matrix?

- c) **(2 points)** How many blocks does the cache contain?

- d) **(6 points)** How many compulsory misses occur?

- e) **(4 points)** How many capacity misses occur?

- f) **(2 points)** How many conflict misses occur?

- g) **(6 points)** Would a loop interchange decrease the number of capacity misses? If yes, rewrite the loops with the loop interchange (**no need** to repeat the swapping instructions) and provide the new number of capacity misses.
- h) **(2 points)** What would be the number of compulsory misses with a loop interchange?
- i) **(4 points)** What should be the minimal size of the cache if we wanted to use blocking?

B) Virtual Memory (15 points)

We consider a 32-bit address bus and a 2 MB physical memory. Page size is 32 KB. Assuming a validity bit and a dirty bit, the objective is to compute the size of the table.

a) **(2 points)** How many entries does the page table have?

b) **(2 points)** How many frames are in the main memory?

c) **(4 points)** What is the size of each entry (page table)?

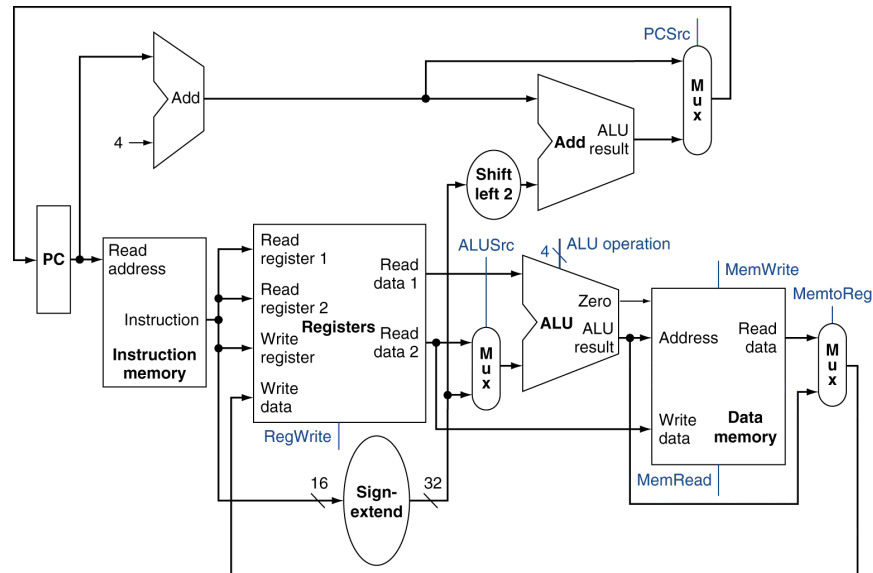
d) **(3 points)** What is then the size of the page table?

e) **(4 points)** Suppose that physical memory is managed like direct-mapped cache: each page with number i is stored in the frame with number $n \% 2^6$. What is in hexadecimal the physical address corresponding to the virtual (logical address) 0x567489?

Exercise 2) (20 points)

Consider the datapath on this figure:

Full Datapath



Chapter 4 — The Processor — 20

- 1) (2 points) Consider the following instruction:

subw \$r1, Constant(\$r2) \$r1 <- Constant - \$r2 (Constant is 16 bits wide)

Example: if \$r2 = 8 and Constant = 120, the CPU will compute the difference (120 - 8) and will store it in Register \$r1, i.e \$r1 = 112 after the execution of the instruction.

Propose an instruction format for **subw** (consistent with the instruction set defined so far. This means that the first 6 most significant bits are reserved for the opcode).

Opcode	
--------	--

- 2) (1 point) Does the full datapath support the **subw** instruction? **Answer only Yes or No. (No justification is needed here).** If the answer is Yes, jump to Question 4.
- 3) (2 points) If the answer to Question 2) is **NO**, add **on the figure** all **needed** lines, units, resources, inputs, outputs, multiplexers.... to support the instruction **subw**.
- 4) (6 points) Based on the instruction code you proposed, **draw/highlight on the figure** the datapath used by the instruction **subw**.

- 5) **(8 points)** After you draw the datapath, label the relevant buses with size and the content for the instruction **subw \$3, 250(\$23)**. Assume that \$3 = 32, \$23 = 48.
- 6) **(1 point)** Provide the final values of the registers \$3 and \$25.

Exercise 3: (15 points) Deriving the pipeline speed up

Consider a monocycle CPU with 4 operations (stages) that take, 200ps, 200ps, 200ps, and 400ps.

- 1) **(1 point)** What is the latency of one instruction on the monocycle CPU?
- 2) **(1 point)** Consider a program P with n instructions. What is the execution time for Program P (expression as a function of n) on the monocycle CPU?
- 3) **(11 points)** We want to pipeline the CPU above with one stage per operation. We neglect the buffer time between stages. All the questions below apply to the pipeline
 - a) **(3 points)** What should be the clock frequency for the pipelined CPU?
 - b) **(3 points)** What is the latency for one instruction for the pipelined CPU?
 - c) **(2 point)** What is the throughput (bandwidth) in MIPS of the pipelined CPU?
 - d) **(3 points)** Consider a program P with n instruction. What is the execution time for Program P? (Make sure to take into account the phase to fill up the pipeline and wind it up)
- 4) **(2 points)** What is the expression of the speed up (monocycle versus pipelined)? If n tends to infinity, what is the speed up?