Fall 2016: COMP 7300 Advanced Computer Architecture

Test 2 (100 pts)

Grading policy:

¼ Credit for correct answer only

¾ 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) **(10 points)** We consider a 4-way associative 32 KB cache with 2 KB blocks. The CPU generates the address 0x6795A3.

   a. **(3 points)** What is the block address (in Hexa **OR** binary) of Block ***B*** containing address 0x6795A3?

   2KB block → 2^{11} bytes → offset ~~block~~ bits = 11 bits

   0110 0111 1001 0101 10100011

   block addr ; offset.

   b. **(4 points)** What is the index where Block ***B*** will be stored in the cache?

   32/2 = 16 blocks → 4-way ⇒ 16/4 = 4 sets ⇒ index bits = 2 bits

   So (10) is the index (~~01~~) (01 in binary)

   c. **(3 points)** What is Block B's tag (in Hexa **OR** binary?

   01100111100

(25 points) Virtual Memory: We consider a 32 bit address bus and a 512 KB physical memory. Page size is 16 KB. Assuming a validity bit, a dirty bit, and **one** reference bit.

a) **(3 points)** How many pages does the logical space have?

$2^{32}/16KB = 2^{32}/2^{14} = 2^{18}$ pages

b) **(3 points)** How many entries does the page table have?

# of entries = # of pages = 2^{18} entries

c) **(3 points)** How many frames does the physical space have?

$$\text{\# of frames} = \frac{512KB}{16KB} = 32 \text{ frames} = 2^5 \text{ frames}$$

d) **(4 points)** What is (in Kbytes) the size of the page table?

entrie size: 1 (validity) + 1 (dirty bit) + 1 (reference) + 5 = 8 bits

# of entries: 2^{18}

table size = $2^{18} \times 8 = 2^{21}$ bits

= 256 Kbytes

e) Consider the partial page table below starting at entry 0. **Pay attention!!!**: the values in the page table **INCLUDE** (as most significant bits) the validity bit, the dirty bit, and the reference bit, respectively.

0	0x54
1	0x83
2	0x91
3	0xB2
4	0x34
5	0x89
6	0x77
7	0xE3
.....	.....

0101 0100
1000 0011
1001 0001
0011 0100

a. **(12 points a+b)** The CPU generates the address 0x10854. Is this address is in the main memory? If yes, what is its physical address? # of offset bits = 14 bits

0001 0000 1000 0101 0100

4 pts

entry number = 4 validity bit = 0, not in main memory

b. The CPU generates the address 0x09874. Is this address is in the main memory? If yes, what is its physical address?

0000 1001 1000 0111 0100

8 pts

entry number = 2, validity bit = 1, it is in the main memory

physical address = 1000 1000 0111 0100

Exercise 1: (30 points) We assume that the data cache size is 16 Kbytes (data) and the block size is 128 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 manipulation a matrix of **short** integers (2 bytes per short):

```
short A[128][128];
int i,j;
for (j = 0; j < 64; j++)
      for (i = 0; i < 128; 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];
}
```

A[i][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=127) are swapped, the second column (j=1) and the column (j = 126) are swapped, and so on…

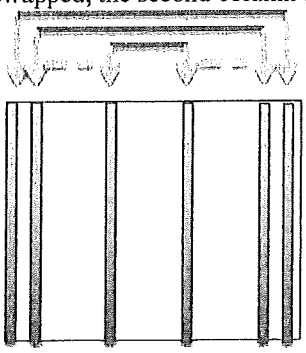


Figure 2

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

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

one line $= 128 \times 2 = 256$ bytes

$256 / 128 = 2$ blocks

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

$\frac{128 \times 128 \times 2}{128} =$ 256 blocks

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

$\frac{16\,KB}{128\ bytes} = \frac{2^{14}}{2^7} = 2^7$ blocks

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

256 blocks needed

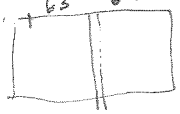
$\Rightarrow$ 256 compulsory misses

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

One swap $\Rightarrow$ 2 capacity misses (after 128 compulsory misses)

There should be 64×128 swaps in total.

$2 \times 64 \times 128 - 128 \times 2 = 2 \times (63 \times 128) = 16128$ capacity misses



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

No conflict misses for Fully associative

g) **(5 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.

Yes,

```
for ( int i=0 ;i<128 ;i++) {
    for (j= 0 ; j<64 ; j++)
        swap
    }
```

No capacity miss will happen because once we load a block, we will do all the swap process before it is discarded

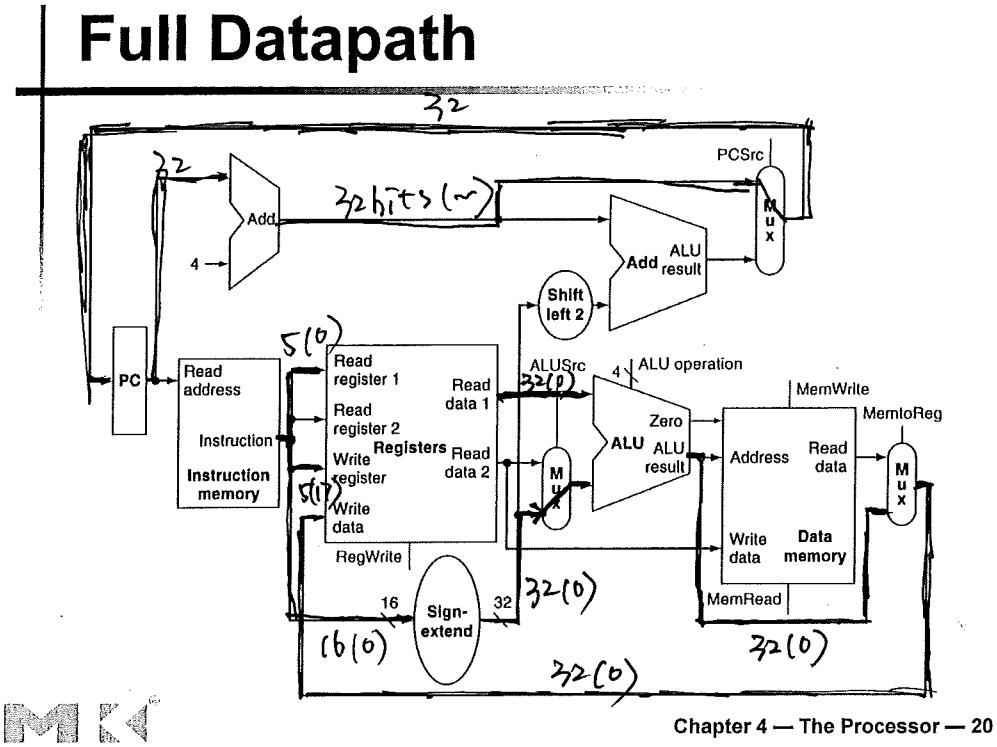
h) **(1 point)** What would be the number of compulsory misses with a loop interchange?

It will not change, 256 compulsory misses

i) **(6 points)** Using loop interchange, what should be the minimal size of the cache to completely eliminate the capacity misses?

The cache should contain 2 blocks at least. because we need to do the swap between 2 blocks, so these 2 blocks must be in cache at the same time, which should be 2 x 128 bytes/block = 256 bytes

Exercise 2) (20 points)
Consider the datapath on this figure:



1) **(3 points)** We propose to implement the ***new*** instruction:

init `$r1` // Set register $r1 to 0
Note that this instruction has no effect on Register $0 because it always contains the value 0.

Example: The CPU will set Register $r1 to 0 ($r1 = 0 after the execution of this new instruction).

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

6 opcode			
Opcode	R0 (read $0)	R1 (write register)	constant (0)
	5 bits	5 bits	16 bits

2) **(1 point)** Does the full datapath support the *new* 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 *new* instruction.

4) **(6 points)** Based on the instruction code you proposed, **draw/highlight on the figure** the datapath used by the *new* instruction.

5) **(8 points)** After you draw the datapath, label the relevant buses with size and the content for the instruction ***init*** *$17*. Provide the values of the control inputs. Assume that $17 = 32.

Programming: (15 points)

Write a program that inputs a 32 bit address (integer) and display the index assuming. We consider a 4-way associative 16 KB cache with 4 KB blocks. Your program must:

1) prompt the user to input an integer (address)
2) compute the index
3) display the index

If you do not remember the system call number of some function, just use any number and comment it.

Grading guidelines: first mistake is forgiven, second **major** mistake costs 2 pts, 3rd (4 pts), 4th (6 pts)....

```
# Test 1 Code
.data
   prompt:      .asciiz "Enter an address : "
   theResult:   .asciiz "\n Index is "


.text
```