CSCI-UA.0201-001

Computer Systems Organization

Final Exam - Fall 2016 (time: 75 minutes)

Last name: First name: NetIT:

(Total of 30 points)

1. (5 points) Circle the correct answer among the choices given. If you circle more than one answer, you will lose the grade of the corresponding question.

(A) Virtual memory gives the illusion of:

a. bigger memory b. faster memory

b. bigger disk d. all of the above

(B) How many times does the following instruction result in level 1 cache access:

movq (%rax), %rbx

a. 1 b.0 c. Depends on the cache size. d.We cannot tell.

(C) All the following cache memories have the same number of blocks, and the same block size. Which of the following has the least number of sets?

a. 4-way set associative b. 2-way set associative

c. Direct mapped d. they all have the same number of sets

(D) Not coalescing blocks after free() in dynamic memory allocator may result in:

a. external fragmentation b. internal fragmentation

c. page fault d. exception

(E) The dynamic memory allocator when implementing malloc() and free() uses:

a. virtual address b. physical address c. depends on the design

2. [8 points] Given the following x86_64 assembly code, fill-in the blanks in the corresponding C code. Also on the far right, fill in the correspondence between each register and its corresponding variable in C. (Hint: you can neglect edx because it does not correspond to any variable in the C code and is used only by the compiler for the translation)

```
foo: xorl %eax, %eax
     movl %edi, %ebx
     sall $1, %ebx
L3: cmpl %edi, %ebx
     jle L0
     testl $1, $ebx
     jne  L1
     addl $2, %eax
    jmp L2
L1: addl $3, %eax
L2: subl $1, %ebx
    jmp L3
L0: ret
```

```
int foo(int x){

   int sum = __0_____;
   int i;

   i =_____x*2_____;

  while (____i > x_____){
       if( i%2 == 0)
        ___sum +=2_____;
        else
        ___sum += 3_____ ;
        i--;
   }
    return sum;
}
```

edi	x
ebx	i
eax	sum

3. [3 points] State three advantages of virtual memory:

- Gives process the illusion it has the whole memory for itself.
- Makes linking and loading much easier
- Protect processes from each other
- Gives the illusion of bigger memory

4. Suppose that we have a system with main memory access time of 100 cycles. We added to that system a cache with 2 cycles access time. The resulting average memory access time becomes 22.

a. [2 points] What is the cache hit rate?

avg mem access latency = mp + (1-p)(M+m) = m + (1-p)M
22 = 2 + (1-p)100
p = 0.8 → 80%

b. [2 points] Did we benefit from having a cache in that case? Why?

yes, without cache avg memory latency = main memory access = 100. With cache it is 22 as indicated above.

c. [1 point] Is the cache mentioned above accessed with virtual address? or physical address?

physical address

d.[1 point] Is the memory mentioned above accessed with virtual address? or physical address?

physical address

5. Given the following cache: each row is a set, the very first row is set 0, and each block consists of valid bit (V), Tag, and data (the block itself). The TAG value shows how many bits we need for the tag. Assume the address length is 32 bits. For this problem, you can neglect the content of the data.

V	TAG	Data	V	Tag	Data
1	0xABCDEF		1	0x123456	
0	0xABCDEF		1	0x56789A	
0	0x56789A		0	0xABCDEF	
1	0x456789		0	0x123456	

a. [1 point] How many bits do we need for the *set* part of the address?

4 sets → 2 bits

b. [2 points] What is the block size? You have all the information you need! So, no assumptions are needed! To get full credit, you HAVE to show all steps.

32 bits → 24 for TA (as shown in the table above) + 2 for sets (as shown in part a above)
→6 bits are left for the block → block size = 2^6 = 64 bytes

c. [1 point] From the content of the cache above, how many blocks currently exist in the cache?

4 (only the blocks with valid bit = 1 counts)

d. [1 point] Show how the 32 bits are divided into TAG, Set, and Index. Show all the steps and include the number of bits in the space below.

TAG	SET	Offset
num bits = 24	num bits = 2	num bits = 6

e. [3 points] For the following address: 0xABCDEF80 do we have a cache hit or miss? if it is a hit, what is the set number? If it is a miss, do we have to use a replacement policy?

The Tag is 24 bits. This means it is 24/4 = 6 hexadecimal digits → 0xABCDEF
The rest of the address is 0x80 → in binary 1000 0000
The two bits from the left are for set → 10 → set# 2
In set 2 from the table above, all blocks have valid bit of 0 → cache miss
No replacement policy needed because the set has room (since it has at least one block of valid bit 0)