

CSCI 2510 Computer Organization 2018-19

Assignment 3

Deadline: November 27, 2018 (TUE) 14:30pm (before the Tutorial session)

Submission Notes:

- (1) For each of the following written exercises (for Question 1-3), please show your steps, and explain in detail when needed to receive full credit.
- (2) Submit two files named CSCI2510_Assignment3.pdf (for Question 1-3) and associative_mapping.asm (for Programming Exercise) to Blackboard Assignment Collection Box before the deadline.
- (3) Late submission per day is subject to 10% of penalty.

Question 1(Virtual Memory 10 pts):

- a) Why the modern operating system uses Virtual Memory? (3 pts)
- b) What is the component that maintains the virtual-to-physical address translation information for each page? (2 pts)
- c) What is the component inside a processor that stores the starting address of the page table? (2 pts)
- b) How to translate the virtual address to physical address? (Hint: You need to think about how page table, page table base register, virtual address, physical address work together.) (3 pts)

Question 2(Basic Processing Unit 20 pts):

Please write down the sequences of steps for the execution: SUB R1, [R2].

Question 3(Cache in Action 20 pts):

A computer system uses 32-bit memory addresses and it has a main memory consisting of 1G bytes. It has a 4K-byte cache organized in the block-set-associative manner, with 4 blocks per set and 64 bytes per block.

- a) Calculate the number of bits in each of the Tag, Set, and Word fields of the memory address.
- b) Assume that the cache is initially empty. Suppose that the processor fetches 1088 words of four bytes each from successive word locations starting at location 0. It then repeats this fetch sequence nine more times. If the cache is 10 times faster than the memory, estimate the improvement factor resulting from the use of the cache. Assume that the LRU algorithm is used for block replacement.

Programming Exercise (50 pts):

Write codes in five labels(check, hit, replace, findfirst, recordfirst) in the associative_mapping.asm to implement the associative mapping using FIFO algorithm.

```
.code
start:
    mov ESI, offset time
    mov EBP, offset cacheBlocks
    invoke crt_printf, addr inputCacheStatement
    invoke crt_scanf, addr inputFormat, addr cacheSize
    mov ECX, cacheSize
    cmp ECX, -1
    je exitprogram
    jmp input

input:
    invoke crt_printf, addr inputStatement
    invoke crt_scanf, addr inputCPUAccessFormat, addr CPUAccess
    add count, 1
    mov ECX, CPUAccess
    cmp ECX, -1
    je exitprogram
    mov EAX, 0 ; count
    jmp check

check:
    fill here

hit:
    fill here

replace:
    fill here

findfirst:
    fill here

recordfirst:
    fill here

printcachestatus:
    mov EDI, 0
    invoke crt_printf, addr stateFormat
    call printstate
    invoke crt_printf, addr countFormat
    mov EDI, 0
    call printcount
    invoke crt_printf, addr endFormat
    jmp input

printstate:
    mov EAX, [EBP + EDI*4]
    invoke crt_printf, addr outputFormat, EAX
    add EDI, 1
    cmp EDI, cacheSize
    jne printstate
    ret

printcount:
    mov EAX, [ESI + EDI*4]
    invoke crt_printf, addr outputFormat, EAX
    add EDI, 1
    cmp EDI, cacheSize
    jne printcount
    ret

exitprogram:
    invoke ExitProcess, NULL

end start
```

Given the cache size of 4, if you input the sequence 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1, the result should be:

```
C:\WINDOWS\system32\cmd.exe
input cache size(minimum 2, maximum 32) or input -1:exit program):4
CPU Access(input positive number or input -1:exit program):7
cache status 7 -1 -1 -1 count status 1 -1 -1 -1
CPU Access(input positive number or input -1:exit program):0
cache status 7 0 -1 -1 count status 1 2 -1 -1
CPU Access(input positive number or input -1:exit program):1
cache status 7 0 1 -1 count status 1 2 3 -1
CPU Access(input positive number or input -1:exit program):2
cache status 7 0 1 2 count status 1 2 3 4
CPU Access(input positive number or input -1:exit program):0
cache status 7 0 1 2 count status 1 2 3 4
CPU Access(input positive number or input -1:exit program):3
cache status 3 0 1 2 count status 6 2 3 4
CPU Access(input positive number or input -1:exit program):0
cache status 3 0 1 2 count status 6 2 3 4
CPU Access(input positive number or input -1:exit program):4
cache status 3 4 1 2 count status 6 8 3 4
CPU Access(input positive number or input -1:exit program):2
cache status 3 4 1 2 count status 6 8 3 4
CPU Access(input positive number or input -1:exit program):3
cache status 3 4 1 2 count status 6 8 3 4
CPU Access(input positive number or input -1:exit program):0
cache status 3 4 0 2 count status 6 8 11 4
CPU Access(input positive number or input -1:exit program):3
cache status 3 4 0 2 count status 6 8 11 4
CPU Access(input positive number or input -1:exit program):2
cache status 3 4 0 2 count status 6 8 11 4
CPU Access(input positive number or input -1:exit program):1
cache status 3 4 0 1 count status 6 8 11 14
CPU Access(input positive number or input -1:exit program):2
cache status 2 4 0 1 count status 15 8 11 14
CPU Access(input positive number or input -1:exit program):0
cache status 2 4 0 1 count status 15 8 11 14
CPU Access(input positive number or input -1:exit program):1
cache status 2 4 0 1 count status 15 8 11 14
CPU Access(input positive number or input -1:exit program):7
cache status 2 7 0 1 count status 15 18 11 14
CPU Access(input positive number or input -1:exit program):0
cache status 2 7 0 1 count status 15 18 11 14
CPU Access(input positive number or input -1:exit program):1
cache status 2 7 0 1 count status 15 18 11 14
CPU Access(input positive number or input -1:exit program):-1
Press any key to continue . . .
```