Class	CS147, Sec 01
Homework	III
Due Date	Dec 08, 2018 11:59 PM PST
Instructions	1. There are 7 questions with total 100 points. 2. Please create electronic document with your answer. 3. There is no need to include the question itself. However, you **MUST** include question number and sub-part index if any. Example: 5(b) 4. Please create a PDF document **hw3.pdf** and ***upload that in Canvas*** assignment page by the due date. 5. **Please re-check you submission for any logistic errors (empty file, corrupted PDF, and many more) and re-submit if needed. Once grading is started, any file with logistics errors will be given 0 point.** 6. **NO** handwritten (and scanned) document is accepted. You answer must be typed in (and computer drawn if diagram is asked) using word processing software. 7. **NO LATE SUBMISSION.** 8. **Please explain your answer clearly – just writing the final answer in a word or two is not sufficient in most of the cases.**

1. Assume a word addressable memory and a register file on a computing system contains the following data at an instance of time during running of a program. What is the operand value retrieved if : [**5*5pts** = **25pts**]
    a. Next instruction is direct addressing with memory address 0x10003010?
    b. Next instruction is indirect addressing with memory address 0x10003010?
    c. Next instruction is register addressing with register address 7?
    d. Next instruction is register indirect addressing with register address 7?
    e. Next instruction is displacement addressing with register address 20 and offset address as 0xF6 (8-bit 2's complement format for offset)?

MEM ADDR	CONTENT
0x1000301B	0x00000005
0x1000301A	0x00000006
0x10003019	0x00000007
0x10003018	0x00000008
0x10003017	0x00000009
0x10003016	0x00000010
0x10003015	0x00000011
0x10003014	0x00000012
0x10003013	0x00000010
0x10003012	0x00000011
0x10003011	0x00000012
0x10003010	0x10003014
0x1000300F	0x10003015
0x1000300E	0x00000001
0x1000300D	0x00000002
0x1000300C	0x00000003

REG	CONTENT	REG	CONTENT
R00	0x00000000	R16	0x90003024
R01	0x00000001	R17	0x90003025
R02	0x00000002	R18	0x90003026
R03	0x00000003	R19	0x90003027
R04	0x00000004	R20	0x10003018
R05	0x00000005	R21	0x10003027
R06	0x00000006	R22	0x10003028
R07	0x10003013	R23	0x10003029
R08	0x90003024	R24	0x10003030
R09	0x00000006	R25	0x10003031
R10	0x00000007	R26	0x10003032
R11	0x00000008	R27	0x10003033
R12	0x00000009	R28	0x10003034
R13	0x00000010	R29	0x10003035
R14	0x00000011	R30	0x10003036
R15	0x00000012	R31	0x10003037

ANS:

a) M[0x10003010] = 0x10003014

b) M[M[0x10003010]] = M[0x10003014] = 0x00000012

c) R[7] = 0x10003013

d) M[R[7]] = M[0x10003013] = 0x00000010

e) M[R[20] + 32-bit Sign Extension of 0xF6] = M[0x10003018 + (-1010)] = M[0x1000300E] = 0x00000001

2. A NUMA computer cluster has 200 computing nodes with each node associated with 128GB RAM. Average local memory access time is 50ns and average remote memory access time is 180ns. Assume a program is running on one node in this cluster. It is consuming total of 230GB with local memory occupancy as 100GB. If there are 1 million **(1,000,000)** memory access in this program with probability of accessing a remote memory location as 0.25 what is the total time the program is busy accessing memory in millisecond (ms) unit ? [**10pts**]

ANS:

Since probability of remote memory location access for this program is 0.25

- Number of access to remote memory location N_{RM} = (0.25*1,000,000) = 250,000
- Number of access to local memory location N_{LM} = (1,000,000 – 250,000) = 750,000

Therefore:

- Total time to access remote memory AT_{RM} = (250,000 * 180) ns = 45,000,000 ns = 45ms
- Total time to access local memory AT_{LM} = (750,000 * 50) ns = 37,500,000 ns = 37.5ms

Hence total time that this program was busy in accessing memory (45+37.5)ms = 82.5ms

3. A computing system with traditional disk (rotating hard disk) transfer data to main memory at minimum unit of 2KB of block of data (i.e. it has to adjust read/write head per 2KB block data access). Its average access time is 1ms. It has transfer rate of 100Mbps (mega-bits per second). Another computing system has solid state disk (with random access technique) transfer data to main memory at minimum unit of 1KB of block of data. It has access time of 50us per block of data transfer and refresh time of 250us. Both the system runs Linux operating system with initial size of 2MB to be loaded into main memory during boot time. Which system will boot faster and how much (ratio their performance)? [**10pts**]

ANS:

- For the system with traditional disk
  - Total number of blocks to be transferred N = (2MB/2KB) = $(2^{20}/2^{10}) = 2^{10} = 1024$
  - Each block will incur one access. Therefore total access time incur will be (1024*1ms) = 1024ms. = 1.024 sec.
  - Each block will transferred to main memory in (2KB/100Mb)sec=((2*8)Kb/100Mb)sec= $(0.16 * 2^{-10})$ sec
  - Hence time to transfer 1024 blocks will be $(0.16 * 2^{-10} * 1024)$ sec = 0.16 sec.
  - Therefore total time for booting up will be (0.16 sec + 1.024 sec) = 1.184 sec.
- For the system with solid state disk.
  - 1KB block transfer [Access + Refresh] will take total (50 us + 250 us) = 300 us
    - One can argue if last read needs refresh – but lets not get into that complexity. Moreover, in practical situation there will be a power up time involved for such memory which can be accounted into last refresh time.
  - Total number of blocks to be transferred N= (2MB/1KB) = $(2*2^{20}/2^{10}) = 2*2^{10} = 2*1024 = 2048$.
  - Therefore total time for booting up will be (2048 * 300us) = 0.6144 sec.

Therefore system with solid state disk will boot faster by (1.184s / 0.6144s) = 1.93 times

4. Your Project II processor is modified to hook into a cache memory system (i.e. states are introduced to handle cache miss) and running on 1GHz clock. This system is connected to a memory system with separate instruction and data cache. Both these caches has access time 1ns and cache miss penalty of 150ns. However, miss rate for instruction cache is 5% and miss rate for data cache is 10%. If the following piece of code is executed in that processor and the 'loop' block (starting from 'loop' label to last 'jmp loop') has been executed 10,000 times what is the total execution time of this program so far in 'us' (microsecond) unit? [**10pts**]

```
        addi r2, r2, 0x0001;
        lui r0, 0x0100;
        sw r1, r0, 0x0000;
loop:   addi r0, r0, 0x0001;
        sw r2, r0, 0x0000;
        addi r3, r2, 0x0000;
        add r2, r2, r1;
        addi r1, r3, 0x0000;
        j  loop;
```

ANS:

Total run time of the program execution has several components.

- Total number of instructions will be executed under this condition is (6*10,000 + 3) = 60,003. Each instruction takes 5 clock cycle to be completed without any stall. Each clock cycle is (1/1GHz)sec long, i.e. cycle time is 1ns. Therefore, without any stall, it will take (60,003 * 5)ns = 300.02us.
- Both cache access time is 1ns. Therefore any memory operation (instruction fetch and memory access in lw/sw instruction) will be perfectly align with the 5-stage operation. Hence no need to account for extra clock cycle for no cache miss scenario.
- However, out of 60,003 instruction fetch 5% will cause cache miss and will incur additional penalty. Therefore total penalty in instruction fetch is (60,003 * 0.05 * 150)ns = 450.02us.
- There will be 10,001 data memory access (10,000 within loop and 1 outside loop). Out of these 10,001 data memory access 10% will incur cache miss penalty. Therefore total data cache miss penalty will be (10,001 * 0.1 * 150)ns = 150.02us.
- Hence, total execution time with 10,000 time loop will be (300.02 + 450.02 + 150.02) = 900.06us

5. A 32-bit computer system supports 4GB byte addressable memory. It's processor has cache with block size of 1KB. Both the control and tag bits per cache line is 8 bits each. Assume that cache line is fully mapped from bits of the address A excluding block bits and tag bits in the address.
    a. How much data is cached within this cache memory in MB unit? [**10pts**]
    b. Also determine actual size of the cache memory in KB unit. [**10pts**]
    c. Whole memory is divided into how many blocks? [**5pts**]

ANS:

a) Total number of bits in address is 32. Out of these 32 bits

- Total bits needed for block index is $\log_2(1K) = \log_2(2^{10}) = 10$ (since block size is 1KB).
- There are 8-bits to indicate tag in an address value.
- Hence (32 – 10 – 8) bits = 14 bits are left for index into cache line.
- Since all cache index bits are fully mapped, there will be 2^{14} = 16,384 cache lines (or 16K cache line).
- Each cache line caches 1KB data, hence total amount of data cached is (16K * 1KB) = 16MB.

b) Apart from 16MB cached data, there are extra 16-bits (2bytes) are stored per cache line. These extra bits will need total (16K * 2B) = 32KB space. Hence actual size of the cache memory is (16MB + 32KB) = (16,384 + 32) KB = 16,416KB.

c) Total memory is 4GB. Since block size is 1KB, there are (4GB / 1KB) = $(4*2^{30}) / 2^{10} = (4*2^{20})$= 4M block (4,194,304).

6. Now, assume that the system in the #5 uses 8-way associative cache but cache same amount of information.
    a. Determine TAG, CACHE LINE INDEX and BLOCK INDEX of the address 0x30AB23F2. [**5pts**]
    b. What is the actual size of the cache memory in this case in KB unit? [**5pts**]

NS:

a) For an 8-way associative there will be 8-sets with each (16K / 8) = 2K cache lines (since same amount of data, 16MB, as #5 is cached). Hence cache line index will need $\log_2(2K) = \log_2(2*2^{10}) = 11$ bits. Since block size is fixed to 1KB with block index size 10-bits, tags for this new cache will be (32 – 11 – 10) = 11 bits.

For given address 0x30AB23F2 binary pattern is

0011 0000 101 | 0 1011 0010 00 | 11 1111 0010

Grouping them into different parts:

- Block index : 0x3F2
- Cache line index : 0x2C8
- Tag : 0x185

b) Cached data amount will remain same as #5. However, tag will now take 11-bits compare to 8-bits in #5. Total number of bits per cache line is (8+11) = 19. For 16K lines (since each set will have this extra 19-bits) will have (19 *16K) bits = 304Kb = 38KB. Therefore total cache size will be (16MB + 38KB) = 16,422KB

7. A 32-bit computer system uses virtual memory with 4K page size. It also uses a cache same as #6. For any memory access, both TLB and cache access time is 1ns each. TLB miss rate is 25% with 20us penalty. Cache miss rate is 10% with 150us penalty. Page fault rate is 0.01% with page service (bring back page from disk to memory) 1s.
    a. How much time would be spent for one million (1,000,000) memory access in second unit? [**5pts**]
    b. Determine TAG, CACHE LINE INDEX and BLOCK INDEX for a virtual address 0x4CD67821 with TLB entry for 0x4CD67 is 0x28701, if this system uses same cache as in #6. [**5pts**]

ANS:

a) There are multiple component of time spent for these 1,000,000 memory access (accordingly the data access model that we have learned in class).

- If there are no TLB and cache miss, each access will take 2ns total (1ns for TLB access for virtual address to physical address translation. Another 1ns for information access from cache.) Therefore, without any miss 1,000,000 memory access will take 2,000,000ns = 0.002s.
- In case of no TLB miss, but cache miss there will be 150us penalty. Since 10% is cache miss rate, therefore total (1,000,000 * 0.1 * 150us) = 15s.
- In case of TLB miss but no page fault, there will be 20us penalty. Since TLB miss rate is 25%, total TLB miss penalty without page fault will be (1,000,000 * 0.25 * 20us) = 5s.
- 0.01% of TLB miss will cause page fault. Hence total number of page fault is (1,000,000 * 0.25 * 0.0001) = 25. Since each page fault is associated with 1s page service time, it will incur total 25s.
- Hence total time, including all the miss and fault handling, for accessing 1,000,000 memory will be (25 + 5 + 15 + 0.002)s = 45.002sec.

Here is the assumption - 10% cache miss already includes the cache miss related to page fault. Note that if it is a page fault, it will cause a cache miss (simply because page fault means the information is not in main memory – if so, the information cannot be at cache.)

b) Since 4K page size, number of page index bit will be 12. Therefore, number of bits to indicate page number will be 20. For a virtual address 0x4CD67821, page number will be 0x4CD67.

Now this 0x4CD67 is mapped to 0x28701. Therefore, corresponding physical address is 0x28701821. Putting this in binary will be as following.

0010 1000 0111 0000 0001 1000 0010 0001

Since this system is using a cache same in #6, bit allocations are as following from MSB:

- Tag : 11-bit
- Cache Line Index : 11-bit
- Block Index : 10-bit

0010 1000 011 | 1 0000 0001 10 | 00 0010 0001

Hence:

- Tag is 0x143
- Cache line index is 0x406
- Block index is 0x021