COS/ELE 375 Assignment 4

Fall 2015, Princeton University
Due date: 1/6/16

Submission guidelines:

Your course enrollment (COS/ELE 375) and date of submission should be listed with your name on the front page of your submission. Please submit your assignment in class or in the box outside of CS-241.
LATE SUBMISSIONS: No late days for HW4.

Collaboration Policy:

You can collaborate on your project team basis. Each team only needs to submit one homework with all the 4 or 5 names on it!

Problems:

1. Using ILP optimization discussed in class, triple the performance of the following loop, or explain why it is not possible. The machine can only do one branch per cycle, but has infinite resources otherwise.

```
        r1 = ... ; r1 is head pointer to a linked list
        r3 = 0

LOOP:
        r2 = M[r1 + 8]
        r3 = r3 + r2
        r1 = M[r1]
        branch r1 != 0, LOOP

        ... = r3 ; r3 is used when loop completes
```

2. Describe the general characteristics of a program that would exhibit very little temporal and spatial locality with regard to instruction fetches. Provide an example of such a program (pseudo-code is fine). Also, describe the cache effects of excessive unrolling. Use the terms static instructions and dynamic instructions in your description.

3. Tomasulo's algorithm enables the processor to dynamically unroll and register rename program code. In terms of hardware resources, what limits the effective amount of unrolling a Tomasulo machine can perform for a given loop? What limits the amount of renaming?

4. Answer the following questions:

   a. What is the average time to read or write a 512-byte sector for a typical disk rotating at 7200 RPM? The advertised average seek time is 8ms, the transfer rate is 20MB/sec, and the controller overhead is 2ms. Assume that the disk is idle so that there is no waiting time.

   b. A program repeatedly performs a three-step process: It reads in a 4-KB block of data from disk, does some processing on that data, and then writes out the result as another 4-KB block elsewhere on the disk. Each block is contiguous and randomly located on a single track on the disk. The disk drive rotates at 7200RPM, has an average seek time of 8ms, and has a transfer rate of 20MB/sec. The controller overhead is 2ms. No other program is using the disk or processor, and there is no overlapping of disk operation with processing. The processing step takes 20 million clock cycles, and the clock rate is 400MHz. What is the overall speed of the system in blocks processed per second assuming no other overhead?

   c. Assume that the bus and memory systems have the following characteristics:
      i. Both the memory and bus system support block accesses of 4 to 16 32-bit words.
      ii. The bus is 64-bit synchronous, clocked at 200 MHz, with each 64-bit transfer taking 1 clock cycle, and 1 clock cycle required to send an address to memory.
      iii. Two clock cycles are needed between each bus operation. (Assume the bus is idle before an access.)
      iv. Assume a memory access time for the first four words of 200ns; each additional set of four words can be read in 20ns. Assume that a bus transfer to the most recently read data and a read of the next four words can be overlapped.

      Find the sustained bandwidth and the latency for a read of 256 words for transfers that use 4-word blocks and for transfers that use 16-word blocks. Also, compute the effective number of bus transactions per second for each case. Recall that a single bus transaction consists of an address transmission followed by data.

   d. Using the disk, memory, and bus described above, if the I/O is allowed to consume 100% of the bus and memory bandwidth, what is the maximum number of simultaneous disk transfers that can be sustained for the two block sizes?

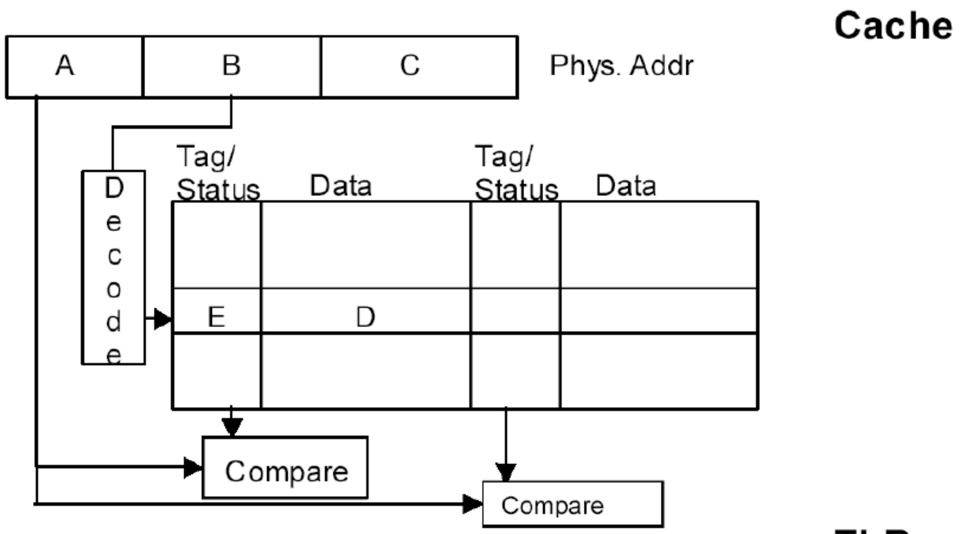
5. Page tables require fairly large amounts of memory, even if most of the entries are invalid. One solution is to use a hierarchy of page tables. The virtual page number can be broken up into two pieces, a “page table number” and a “page table offset.” The page table number can be used to index a first-level page table that provides a physical address for a second-level page table, assuming it resides in memory (if not, a first-level page fault will occur and the page table itself will need to be brought in from disk). The page table offset is used to index into the second-level page table to retrieve the physical page number. One obvious way to arrange such a scheme is to have the second-level page tables occupy exactly one page of memory. Assuming a 32-bit virtual address space with 4-KB pages and 4 bytes per page table entry, how many bytes will each program

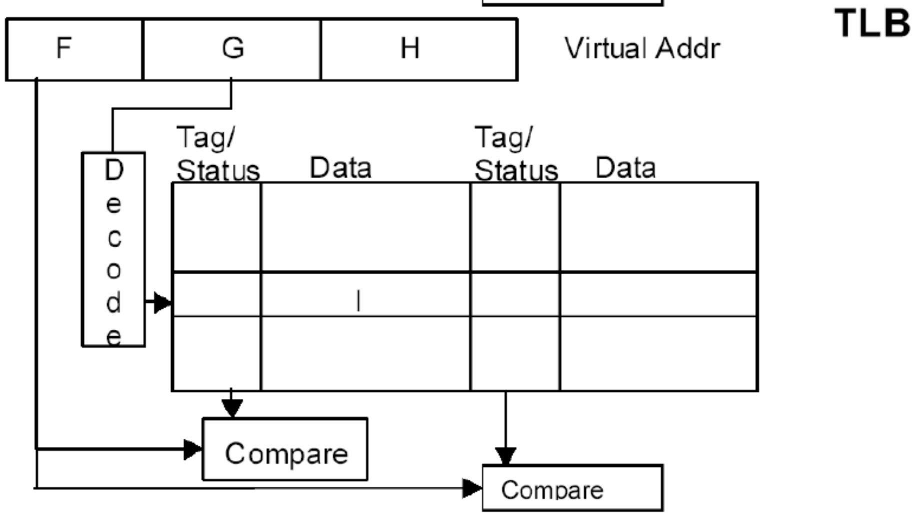
need to use to store the first-level page table (which must always be in memory)? Provide figures that demonstrate your understanding of this idea.

6. Consider a memory system with the following parameters:
   a. Translation Lookaside Buffer has 512 entries and is 2-way set associative.
   b. 64Kbyte L1 Data Cache has 128 byte lines and is also 2-way set associative.
   c. Virtual addresses are 64-bits and physical addresses are 32 bits.
   d. 8KB page size.

   The figures are labeled diagrams of the cache and TLB. Please fill in the appropriate information in the boxes here:

L1 cache		TLB	
A =	bits	F =	bits
B =	bits	G =	bits
C =	bits	H =	bits
D =	bits	I =	bits
E =	bits		


Cache
A
B
C
Phys. Addr
Decode
Tag/ Status
Data
Tag/ Status
Data
E
D
Compare
Compare


TLB
F
G
H
Virtual Addr
Decode
Tag/ Status
Data
Tag/ Status
Data
I
Compare
Compare