Introduction to Computer Architecture

Assignment 2

Due May 18, 2016

1. [50 = 10 x 5] Use the following code fragment:

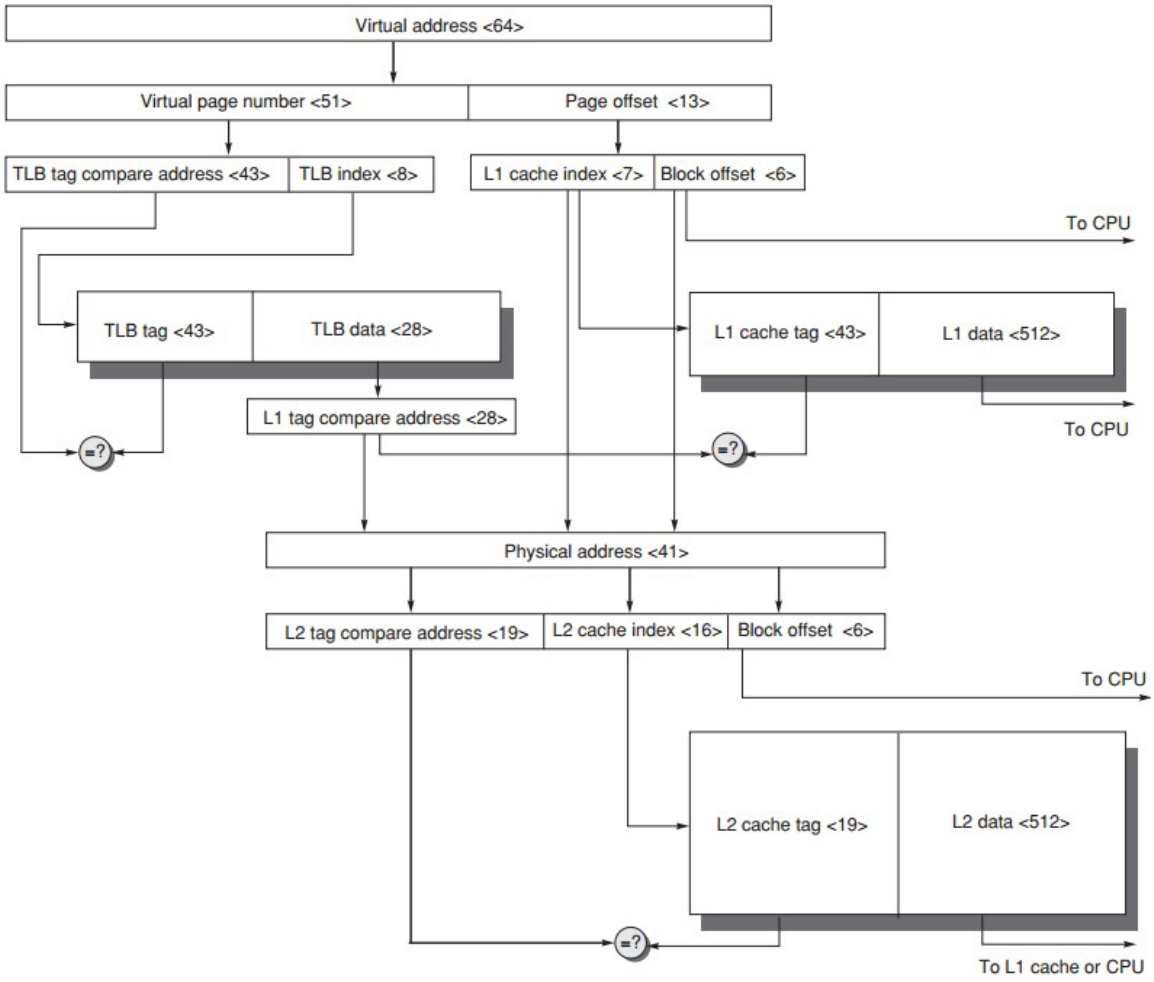
```
Loop:   LD      R1, 0(R2)       ; load R1 from address 0+R2
        DADDI   R1, R1, #1      ; R1 = R1 + 1
        SD      R1, 0, (R2)     ; store R1 at address 0+R2
        DADDI   R2, R2, #4      ; R2 = R2 + 4
        DSUB    R4, R3, R2      ; R4 = R3 - R2
        BNEZ    R4, Loop        ; branch to Loop if R4 != 0
```

Assume that the initial value of R3 is R2 + 396.

a. Data hazards are caused by data dependences in the code. List all of the data dependences in the code above. Record the register, source instruction, and destination instruction; for example, there is a data dependency fro register R1 from the LD to DADDI, that is,
   R1 LD DADDI
b. Show the timing of this instruction sequence for the 5-stage RISC pipeline without any forwarding or bypassing hardware but assuming that a register read and a write in the same clock “forwards” through the register file, as shown in Figure C.6. Use a pipeline timing chart like that in Figure C.5. Assume that the branch is handled by flushing the pipelining. If all memory references take 1 cycle, how many cycles does this loop take to execute?
c. Show the time of this instruction sequence for the 5-state RISC pipeline with full forwarding and bypassing hardware. Use a pipeline timing chart like that shown in Figure C.5. Assume that the branch is handled by predicting it as not taken. If all memory references take 1 cycle, how many clock cycles does this loop take to execute?
d. Show the timing of this instruction sequence for the 5-stage RISC pipeline with full forwarding and bypassing hardware. Use a pipeline timing chart like that shown in Figure C.5. Assume that the branch is handled by predicting it as taken. If all memory references take 1 cycle, how many clock cycles does this loop take to execute?
e. High-performance processors have very deep pipelines-more than 15 stages. Imagine that you have a 10-stage pipeline in which every stage of the 5-stage pipeline has been split in two. The only catch is that, for data forwarding, data are forwarded from the end of a pair of stages to the beginning of the two stages where they are needed. For example, data are forwarded from the output of the second execute stage to the input of the first execute stage, still causing a 1-cycle

delay. Show the timing of this instruction sequence for the 10-stage RISC pipeline with full forwarding and bypassing hardware. Use a pipeline timing chart like that shown in Figure C.5. Assume that the branch is handled by predicting it as taken. If all memory references take 1 cycle, how many cycles does this loop take to execute?

2. [10] Describe the address translation process on the following memory system.



3. [24 = 6 x 4] Virtual machines can lose performance from a number of events, such as the execution of privileged instructions, TLB misses, traps, and I/O. These events are usually handled in system code. Thus, one way of estimating the slowdown when running under a VM is the percentage of application execution time in a system versus user mode. For example, an application spending 10% of its execution in a system mode might slow down by 60% when running on a VM. Figure 2.32 lists the early performance of various system calls under native execution, pure virtualization, and paravirtualization for LMbench using Xen on an Itanium system with times measured in microseconds (courtesy of Matthew Chapman of the University of New South Wales).

a. What types of programs would be expected to have smaller slowdowns when

running under VMs?

b. If slowdowns were linear as a function of system time, given the slowdown above, how much slower would a program spending 20% of its execution in system time be expected to run?

c. What is the median slowdown of the system calls in the table above under pure virtualization and paravirtualization/

d. Which functions in the table above have the largest slowdowns? What do you think the cause of this could be?

Benchmark	Native	Pure	Para
Null call	0.04	0.96	0.50
Null I/O	0.27	6.32	2.91
Stat	1.10	10.69	4.14
Open/close	1.99	20.43	7.71
Install sighandler	0.33	7.34	2.89
Handle signal	1.69	19.26	2.36
Fork	56.00	513.00	164.00
Exec	316.00	2084.00	578.00
Fork + exec sh	1451.00	7790.00	2360.00

4. [16 = 8 x 2] For the code below, assume we have an 8 KB direct-mapped data cache with 16-byte blocks, and it is a write-back cache that does write allocate. The elements of a and b are 8 bytes long since they a double-precision floating-point arrays. There are 3 rows and 100 columns for a and 101 rows and 3 columns for b. Assume they are not in the cache at the start of the program.

Determine the number of cache misses and which accesses cause them for the following codes with or without prefetching.

```
for (i = 0; i < 3; i = i+1)
        for (j = 0; j < 100; j = j+1)
                a[i][j] = b[j][0] * b[j+1][0];
```

```
for (j = 0; j < 100; j = j+1) {
        prefetch(b[j+7][0]);
        /* b(j,0) for 7 iterations later */
        prefetch(a[0][j+7]);
        /* a(0,j) for 7 iterations later */
        a[0][j] = b[j][0] * b[j+1][0];};
for (i = 1; i < 3; i = i+1)
        for (j = 0; j < 100; j = j+1) {
                prefetch(a[i][j+7]);
                /* a(i,j) for +7 iterations */
                a[i][j] = b[j][0] * b[j+1][0];}
```