

Final Exam for ECE 152

name: _____

1) [15 points] A 2-way set-associative cache has 8 frames, and the block size is four bytes. For the following byte addresses, which ones hit and which ones miss? Also, what are the final contents of the cache after this stream of accesses (give the byte address of the first byte in the block)? You may find it helpful to draw a diagram of the cache.

Addresses: 0, 16, 1, 17, 48, 9, 10, 38

2) [20] The IEEE 754 floating point standard specifies that floating point numbers have one sign bit, an 8-bit exponent (with a bias of 127), and a 23-bit significand. Using this format, add the numbers 1.75 and 15.0. Your result must be a normalized IEEE 754 format floating point number.

3) [10] What is the engineering motivation for using a multi-level page table? Do you think that this motivation will disappear in future computer architectures?

4) [25] Write MIPS assembly code for the following C/C++ code. Use the appropriate MIPS conventions for procedure calls, including the passing of arguments and return values, as well as the saving of registers that need to be saved. Assume that there are 4 argument registers (\$a1-\$a4), 2 return value registers (\$v1-\$v2), 10 callee-saved registers (\$s1-\$s10), and 10 temporary caller-saved registers (\$t1-\$t10).

```
int main (){
    int val1, val2, val3, val4, val5;
    return add_five_abs(val1, val2, val3, val4, val5);
}
int add_five_abs (int n1, int n2, int n3, int n4, int n5){
    int result = abs(n1)+abs(n2)+abs(n3)+abs(n4)+abs(n5);
    return result;
}
int abs(int num){
    if (num < 0) return -1*num;
    else return num;
}
```

5) [10] In the following MIPS code snippet, identify all data hazards by drawing an arrow from the register that is written to where it is later read. Assume the same 5-stage pipeline that we used in class. I have drawn one of the data hazard arrows to show you what I'm looking for, but you must draw the rest.

```
add $r7, $r4, $r4
```

```
add $r1, $r0, $r7
```

```
lw $r4, 3($r1)
```

```
add $r3, $r4, $r1
```

```
lw $r3, 2($r4)
```

6) [10] A hot new research topic is something known as “network offload.” The idea is that the processor spends too much of its time processing incoming network packets, so we should design network interface cards (a.k.a. network I/O controllers) that can do the processing themselves. Assume your computer has a 8 Mbit/sec network card, and that it is always continuously receiving data (a simplifying assumption). Assume that your processor can execute 100 million instructions per second. Assume that each incoming network message is 1 KByte, and that it takes your processor 1K instructions to process each incoming message. Explain why you think network offload is a good idea or not. Would it be a better or worse idea for a system with a next generation 1 Gbit/sec network card?

7) [10] In a virtual physical cache (virtually indexed, physically tagged), why must the cache size divided by the cache associativity be less than or equal to the page size?

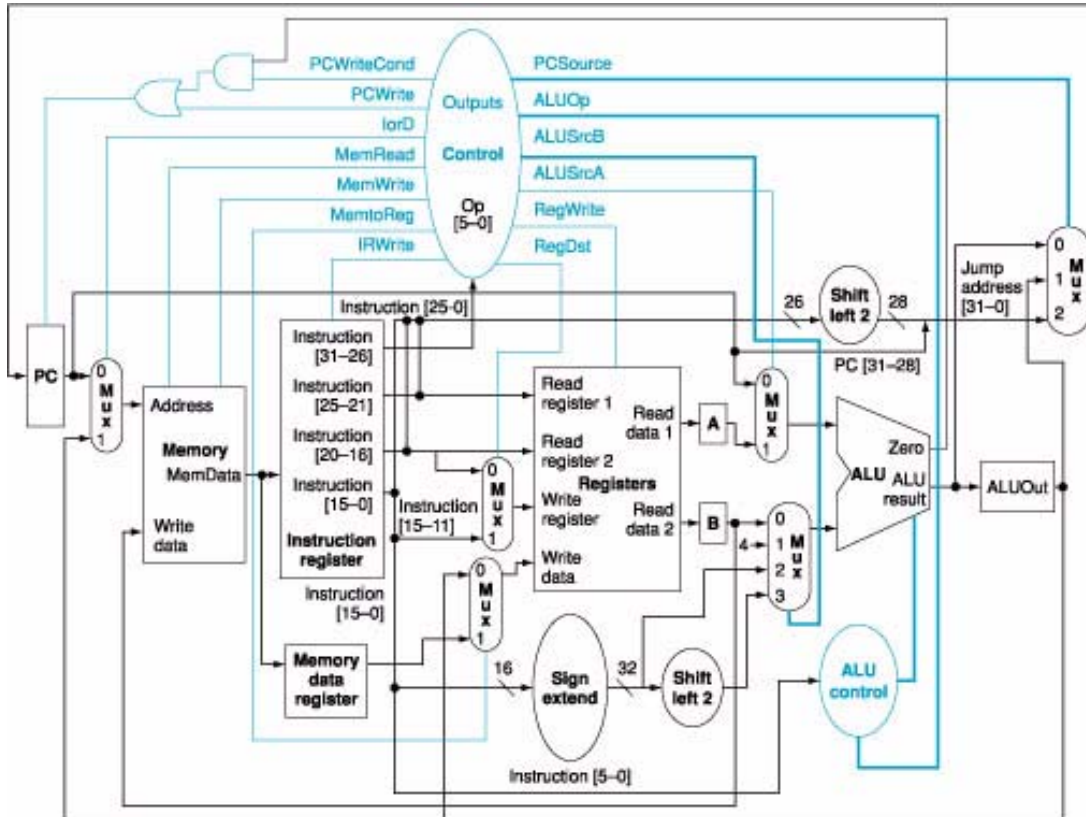


FIGURE 1. Multicycle datapath for Question #8. Note that this is NOT exactly the same datapath that we used in class.

8) [25] Some architectures have instructions that can load a word from a memory address that is specified by Mem[\$rs]. That is, assuming a MIPS-like I-type instruction format, they load Mem[Mem[\$rs]] into register \$rt. How would you add this instruction into the multi-cycle datapath shown in Figure 1?

Discuss what new datapath elements are needed and how control would have to change to accommodate this instruction. You may write on Figure 1, and you may continue your answer on the next page. Please explain your answer thoroughly, so I can more easily give partial credit.

Bonus points [5]: How would this instruction fit into the 5-stage pipeline we studied in class (or in the textbook)? What would have to change to accommodate this instruction in the pipeline?

