CS 230 Practice Final Exam & Actual Take-home Question

Part I: Assembly and Machine Languages (22 pts)

1. Assume that assembly code for the following variable definitions has already been generated (and initialization of A and length).

```
int powerOf2;                  /* powerOf2 = $t0 */
int decimal;                   /* decimal = $t1 */
char A[] = "1011011";          /* A = $s0 */
int  length = strlen (A);      /* length = $t2 */
char * p;                      /* p = $t3 */
```

Write MIPS assembly code for the following code segment, using the registers specified in the comments above (to make it easier for us to go over in class). Include comments. (10 points)

```
powerOf2 = 1;
decimal = 0;
for ( p = A + length - 1; p >= A; p-- )
{
     if ( *p == '1' )
          decimal += powerOf2;
     powerOf2 *= 2;
}
```

```
          addi $t9, $zero, 49      # t9 = 49 ('1')
          addi $t0, $zero, 1       # powerOf2 = 1
          move $t1, $zero          # decimal = 0
          add $t3, $s0, $t2        # p = A + length
LOOP:     addi $t3, $t3, -1        # p--  (orig A + length - 1)
          slt $t5, $t3, $s0        # t5 = 1 if p < A
          bne $t5, $zero, ENDLP    # goto ENDLP if p < A
          lw $t6, 0($t3)           # t6 = *p
          bne $t6, $t9, ENDIF      # goto ENDIF if *p != '1'
          add $t1, $t1, $t0        # decimal += powerOf2
ENDIF:    sll $t0, $t0, 1          # powerOf2 *= 2
          j LOOP                   # goto top of LOOP for p--
ENDLP:    …
```

Could also implement this in other ways!

2. Assemble the following MIPS assembly language code to machine language. Write the machine language instructions in decimal, not in binary. I've done the first line as an example. (8 pts)

```
swap:  add $t1, $a1, $a1        0  5  5  9  0  32
       add $t1, $t1, $t1        0  9  9  9  0  32
       add $t1, $a0, $t1        0  4  9  9  0  32
       lw  $t0, 0($t1)          35 9  8  0
       lw  $t2, 4($t1)          35 9  10 4
       sw  $t2, 0($t1)          43 9  10 0
       sw  $t0, 4($t1)          43 9  8  4
       jr  $ra                  0  31 0  0  0  8
```

3. What is the difference between a *Reduced Instruction Set Computer (RISC)* and a *Complex Instruction Set Computer (CISC)?* What are some benefits and disadvantages of this architectural design? (4 pts)

Part II: Numbers (12 pts)

4. Explain why the IEEE single precision representation for 1 is 3F800000. (4 pts)

00 11 1111 1000 0000 0000 0000 0000 0000 = 0 01111111 00000000000000...

= + 1.0 * 2 ^(127-127) = + 1.0 * 2^0 = 1.0

5. What is the IEEE single precision representation for –2 3/16? (4 pts)

-2 3/16 = 10.0011 = 1.00011 * 2^1 = 1.00011 * 2^(128-127) (so exp = 128)

1 10000000 00011000000... = C 0 0 C 0 0 0 0

6. The way a computer represents real numbers is called *floating point* representation. Why is this term used rather than *real number* representation? (4 pts)

"Real number" implies the full, infinite set of real numbers, but we can only represent a limited number of those in a computer. We cannot accurately or precisely represent all real numbers.

Part III: Logic Gates (10 pts)

7. Show the truth table for a two-input *exclusive-or* function and implement it using AND, OR, and NOT gates. (Reminder: The output of the exclusive-or function is true if and only if exactly one of its inputs is true.) (5 pts)

8. Show the gate diagram for a 1-bit ALU that supports the AND and OR functions. There should be two inputs and one output. Explain how this could be expanded to construct an ALU that acts on a full word. (5 pts)

Part IV: Datapaths and Pipelining (10 pts)

9. Using the datapath diagram provided for the homework assignment, illustrate the data and control paths used by the instruction `lw $s0, 8($sp)`. (8 pts)

10. In a single-cycle datapath, instructions are executed one at a time, one instruction in each clock "tick". In other words, the clock cycle time is set to the time it takes any instruction to fully execute.

    In both a multi-cycle datapath and a pipelined datapath, a clock "tick" is the time an instruction spends in any given stage of the datapath. Each stage has a *latency time* associated with it, which is the amount of time required for the stage to do its job. An example set of stages, with latencies, might be:

    IF: Instruction fetch (200ps)
    ID: Instruction decode and register file read (100ps)
    EX: Execution or address calculation (e.g., time spent in ALU) (200ps)
    MEM: Data memory access (225ps)
    WB: Write back to register file (100ps)

    In a multi-cycle datapath, instructions are executed one at a time, but only go through the stages that they need to. (E.g., R-format instructions do not have to go through MEM, because they don't read or write to memory.) In a pipelined architecture, as each instruction moves to the next stage the following instruction moves into the stage behind it. (8 pts)

    a) A stage's latency describes the minimum amount of time an instruction would have to spend in that stage, but, depending on the datapath type, an instruction might spend longer in a stage than its latency requires. Given the latencies described above, what would be the clock cycle time for a single-cycle datapath? For a multi-cycle datapath? For a pipelined datapath?

    Single-cycle: 825ps (sum of all stages)
    Multi-cycle: 225 ps (slowest stage)
    Pipelined: 225 ps (slowest stage)

    b) How long would a `sw` instruction take in each of these three cases?

    Single-cycle: 825ps (sum of all stages)
    Multi-cycle: 1225 ps (5 * clock cycle because sw requires all stages)
    Pipelined: 1225 ps (5 * clock cycle)

    c) How long would a `beq` instruction take in each of these three cases?

    Single-cycle: 825ps (sum of all stages)
    Multi-cycle: 675 ps (3 * clock cycle because sw requires all stages)
    Pipelined: 1225 ps (5 * clock cycle)

    d) In a pipelined datapath, why does it make sense to require all instructions to go through all the stages of the datapath, whether they are necessary or not? (E.g., R-format instructions have to go through MEM, even though they don't read or write to memory.)

Consider an lw (uses all 5 stages) followed by an add (which doesn't seem to need the MEM stage. They would both be doing their WB at the same time if the add skipped the MEM stage.

e) If we could split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what would be the new clock cycle time of the processor?

Split the slowest stage (MEM). The clock cycle could now be sped up to 200 ps.

CS 230 Practice Final Exam & Actual Take-home Question

11. Describe each of the following hazard types: (6 pts)

structural hazards — hardware-related hazards

control hazard — hazard relating to branching or jumps (start executing one instr, but beq/bne/j elsewhere)

data hazard — need data from prev. instr whose results have not been stored in registers yet

12. Describe two methods for handling hazards: one at the hardware level and one at the software level. (4 pts)

stalls (need to describe them)

no-ops (or nop or bubble) (need to describe them)

(You could also describe fancier software methods, such as re-ordering instructions.)

Part V: Memory (24 pts)

13. Which code fragment is a better example of temporal locality? Which is a better example of spatial locality? Consider data only and not the fetching of instructions. (4 pts)

```
for ( k = 0; k < 100; k++ )
     a[k] = b[k] * 2.5;
```
spatial: goes through A & B in sequential order

```
for ( k = 0; k < y; k++ )
     sum = sum + z;
```
temporal: references sum & z over and over again

14. What are a cache "hit" and a cache "miss"? How are they detected? (4 pts)

hit: the data you're looking for is in the cache
miss: data is not in cache, must go out to memory to retrieve it
detection: the high-order bits of the memory location are compared to the tag in the cache. If they match, the data in the cache is the data you want.

15. Consider a 2-way set-associative 8-line cache with 1 4-byte word per line. Assume, for illustration purposes, that each byte address contains its own value as data, *e.g.*, address 3 contains a 3, address 16 contains a 16, *etc*. Show the evolving cache state for the sequence of (byte) address references below, crossing out values as they are replaced with other values. Circle the references that result in a cache hit:
3, 16, 7, 6, 3, 5, 15, 16, 17, 20, 19, 22, 48 (8 pts)

Binary version of those addresses: (I'm only showing low-order 16 bits of each address for simplicity)

3: 0000 0011 miss
16: 0001 0000 miss
7: 0000 0111 miss
6: 0000 0110 hit (because it was brought in with address 7)
3: 0000 0011 hit
5: 0000 0101 hit (because it was brought in with address 7)
15: 0000 1111 miss
16: 0001 0000 hit
17: 0001 0001 hit (because it was brought in with address 16)
20: 0001 0100 miss
19: 0001 0011 hit (because it was brought in with address 16)
22: 0001 0110 hit (because it was brought in with address 20)
48: 0011 0000 miss

Remember, according to the instructions above, we're pretending that each byte in memory holds the value of its own address; i.e., byte 0 holds int 0, byte 1 holds int 1, byte 33 holds int 33, etc. That way, looking at the data we can see what address it corresponds to.

	v	Tag	00	01	10	11
00	Y	~~0000~~ 0011	~~0~~ 48	~~1~~ 49	~~2~~ 50	~~3~~ 51
	Y	0001	16	17	18	19
01	Y	0000	4	5	6	7
	Y	0001	20	21	22	23
10	N					
	N					
11	Y	0000	12	13	14	15
	N					

16. Describe two replacement algorithms for set-associative memory. (4 pts)

LRU: least-recently-used (need to describe)
Random (need to describe)

alt 16. (From an older practice exam): Describe two write policies for writing from cache to memory. (4 pts)

write-back (need to describe)
write-through (need to describe)

17. What are two reasons for using virtual memory? (4 pts)

We have not talked about this in class.

Part VI: Parallel Processing (8 pts)

18. What is a SIMD parallel processor? (4 pts)

Single-Instruction-Multiple-Data (also called a vector processor): applies the same instruction to multiple data values in parallel.

19. How do the processors in a MIMD distributed memory parallel computer communicate? (4 pts)

We have not talked about this at all in class.