CS61c-FINAL EXAM: 12/11/98 V2 SOLUTIONS

Solutions are slightly different for V1

Question 1 (6 points):

a. Circle the hexadecimal representation for this binary number (including a binary point):
11010001010001110011.0111110 (Hint: in decimal 3.14 is the same as 3.140)
ANS: rewrite as 1 1010 0010 1000 1110 0011.0111 1100
adding a 0 at end, making 1 A 2 8 E 3 . 7 C
Here are your choices:

0x1B28E3.7C 0x1A28E3.7C 0x1A28E3.73 0x1B28E3.1F
or, something else:_____

b. Write this binary number 0101011101110101 in OCTAL (base 8) _____
0101011101110101= 0 101 011 101 110 101 = **53565**
test version 1 had 1 010 111 011 101 010 = **127352**
c. Write the value of each of these binary integer numbers in DECIMAL

Sign magnitude 11011000 =-1*(101 1000) = -58hex = -(5*16+8) = -88

Two's complement 101010000 = - (0 1010 1111 +1) = - B0 hex = -176

Bias-127 00100111 = 0010 0111 = 27 hex = 39 decimal. 39-127 =-88

One's Complement 10100110= - 0101 1001 = 59hex = -89

Question 2 (5 points):

a. How does a RISC design make it easier to build a pipelined implementation of an architecture? Give two reasons (there are certainly more than two possible).

ANS: Here are a few: RISC has fixed length instructions, so it is easy to find the beginning of the next instruction without decoding this one. Limited access to and from memory reduces the number of pipeline stages/stalls to get memory. Forwarding is more likely to work since operations work mostly from registers.

b. A visitor from a distant galaxy lands in Sproul Plaza and hands you a complete computer system that runs the MIPS architecture at 10,000 times the speed of any MIPS computer built on earth. The only problem is that it is missing the SB (store byte) instruction. What do you do to make this valuable? (i) Send it back for repairs (ii) Change all character strings to use 32-bit bytes (iii) Change the MAL assembler (iv) Change the C compiler? Make a choice and explain why, in one or two complete English sentences.

ANS: The simplest change would be to define SB as a MAL op-code and make some appropriate sequence of instructions in TAL that had the same effect. Or define SB via a syscall. In fact, simulating SB is fairly complicated, requiring that you load the word in which the byte is found into a register, clear out the relevant byte by AND, and then OR in the relevant byte, then store into memory. I don't believe it can be done with just the assembler-temporary register. It is important to note that you would have to use LW and SW somewhere. It is possible you could replace the representation of a character by a byte and use a 32 bit quantity, but this would not account for all uses of SB, and we therefore gave partial credit to people who claimed solutions ii, iv. We gave no credit for recommending sending the computer back for repairs on the grounds that this would not make the computer valuable.

Question 3 (17 points):

True/False (v1 is same but renumbered so q is first)

a. __T__Virtual memory conveys the illusion of continuous consecutive memory locations.
b. __F__Virtual memory address space is usually smaller than physical memory space.
c. __T__A very large page size means the TLB will fill up slower.
d. __T__For a given address space, a very small page size means the page table has to be larger.
e. __T__The TLB is the only hardware support on the MIPS R-2000 for paging.
f. __T__If a paging system has no "dirty" bits, they can be simulated by "write-enabled" bits.
g. __F__Branch prediction is primarily an attempt to fix Structural Hazards in a pipeline.
h. __T__Reordering code is a possible way to avoid pipeline stalls.
i. __T__Forwarding is primarily an attempt to fix Data Hazards in a pipeline.
j. __T__Delayed Branch (using a delay slot) is primarily an attempt to fix Control Hazards in a pipeline.
k. __F__After the sixth sequential instruction is started, a five stage pipeline always has five instructions in execution.
l. __F__The Java Virtual Machine (JVM) operation ILOAD_2 loads an integer into register 2.
m. __F__The format of the JVM class file is different for Solaris Intel and HP machines.
n. __T__ Providing cache coherence via write-invalidate allows only a single writer among shared memory multiprocessor systems.
o. __T__One common kind of SIMD computer design is to have vector registers.
p. __F__ If program P uses 50% of its time doing floating-point arithmetic, then it is possible we can make P run more than twice as fast by pipelining floating-point arithmetic.
q. __T__ If program Q spends 90% of its time on loads and stores to memory, then it is possible we can make Q run more than twice as fast by rearranging the storage layout.

Question 4. (15 points):

ANS: (as given, though we took some variations in names, forms, etc.)

Consider the following data declaration section in a MAL program

```
            .data
label1:     .word label2
```

```
            .word 1   #or 11 in v1
label2:     .word label3
            .word 19  #or 9 in v1
label3:     .word 0
            .word 2
```

This is a representation of **three links** in a singly-linked list.

a. Write a `C struct` which corresponds to a unit of that data structure.
The order of the struct's components is important.

ANS (2 pts):

```
struct lnode {
      struct lnode* next;
      int data;
};
```

b. Write a short C program using the structure of part a, to create the data above. Note that in C you cannot do all the initialization in the "data" or declaration section as we have done in MAL.

ANS(3 pts): `struct lnode a, b, c;`

```
a.next = &b;      a.data = 1;  /* or 11, 9 in v 1 */
b.next = &c;      b.data = 19;
c.next = 0; /* NULL */
c.data = 2;
```

c. Write a brief C procedure or code loop to traverse the data structure in C using `printf` to print the data. Assume the code you wrote in part b is correct and that you can make use of it in this part (c). Don't worry if you are unsure of the syntax.

ANS (5 pts):

```
struct lnode* t;
      for (t = &a; t; t = t->next) {
       printf ("%d\n", t->data);
      }
```

d. Traverse the data structure in MAL using the mythical "instruction" "puti $r", which prints the value in register $r as a decimal number. Do not write a function, just a few lines of code. You do not need to write a proper prologue and epilogue. Just assume those parts have been written for you.

Assume you are making use of the MAL code given at the top of this question.

```
# $s0 = t

    loopinit:  la     $s0, label1
    looptop:
    looptest:  beqz   $s0, loopexit

               lw     $a0, 4($s0)
```

```
            puti    $a0

loopbottom:         lw      $s0, 0($s0)
            j       looptop

loopexit:
```

Question 5. (8 points):

Consider the following code:

```
int array[256];
int i, j;
for (i = 0; i < 100; i++)
  for (j = 0; j < 256; j += STRIDE)
    array[j]++;
```

Our choices of STRIDE are 1, 2, and 3. For each of the following scenarios, pick which choice of stride will result the fastest Average Memory Access Time (AMAT), and which will result in the slowest. If there is a tie, you may have more than one answer. On this machine a word is the same size as an int, and there are separate data and instruction caches.

a. Fastest: __1,2,3__Slowest: _1,2,3____Cache size 256 words, cache width 1 word, direct-mapped (All of array is in the cache.. you must realize it is a TIE…)

b. Fastest__3____Slowest:__1 or 2___Cache size 128 words, cache width 1 word, direct-mapped
(only half the array fits in the cache; accessing all the elements is therefore slower)

c. Fastest:___2 or 3__Slowest:___1 ____Cache size 128 words, cache width 1 word, 2-way associative

d. Fastest:__1__ Slowest:___3__ Cache size 32 words, cache width 4 words, direct-mapped

Question 6. (2 points):

Many floating-point numbers can be converted exactly from double-precision to single-precision and back. What can you say about the fraction and exponents of these numbers?

ANS: these are exactly the fraction and exponents that can be represented exactly in single precision, 23 bits and 8 bits.

Question 7. (12 points):

See iout.s, 24 items, ½ point each We are not reproducing the answer here. See your notes. Most people got this entirely right, too!

Question 8. (4 points):

Using only AND gates, OR gates (2 inputs to each gate only), and inverters, draw a logic circuit that takes three input bits A, B, and C, and produces an output that is the same as the majority of the three inputs. That is, if there are two or more 1's, the result is 1. Otherwise (there are two or more 0's) , the result is 0. Fill in this truth table first:

A	B	C	result
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Results should be 0 0 0 1 0 1 1. (1 point) There are various ways of arranging this; the most direct in my mind was to compute s1=A and B, s2= A and C, s3= B and C. Then compute or(s1,or(s2,s3)) to give majority.

Question 9. (4 points):

We were told that virtual memory can give each program the illusion of being the only one in memory. Somehow that doesn't seem to make the loading/link process (merging a.o, b.o, etc. into a single binary executable) unnecessary. Explain some services that are still necessary in loading/linking, even in a virtual memory system.

Ans: At least 2 services. For example, allowing variable names to be external to a .o file; allowing subroutine names (external entry points) to be used; loading two or more .o files in the same (virtual) address space.