Computer Architecture: Spring 2001 – Test 1 Extra Credit Solutions

The first 5 questions are worth 1 pt each.
Assume the following memory contents (all values in HEX). For questions 1-5, assume the MIPS processor is operating in LITTLE ENDIAN mode for memory accesses (same as Intel processors).

Address	Contents															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00001000	C5	67	A5	00	12	BC	34	BB	F4	72	09	A3	29	01	D4	CE
00001010	FE	89	02	D8	A4	8A	7C	DD	90	3C	65	83	9B	19	F6	8A
00001020	A7	CC	9A	BD	8E	90	2C	00	1C	90	0E	13	8C	39	58	C6
00001030	76	D7	CA	FF	D8	71	18	24	40	A8	2C	76	93	C5	0F	9E
00001040	82	A6	54	2E	9A	20	0A	98	E4	A0	0E	25	38	29	2C	86

1. Assume register $8 has the value 0x00001018. What does the register '$9' end up with after execution of the following instruction:

   lw $9, 4($8) *4 + $8 = 0x0000101C, $9 = 0x8AF6199B*

2. Assume register $8 has the value 0x00001018. What does the register '$9' end up with after execution of the following instruction?

   lb $9, 4($8) 4 + $8 = 0x0000101C, *$9 = 0xFFFFFF9B ('lb' sign extends!!!!!, lbu does not sign extend).*

3. Assume register $8 has the value 0x00001018. What does the register '$9' end up with after execution of the following instruction:

   lw $9, 3($8) *3 + $8 = 0x0000101B, MIS-ALIGNED access, cannot do a word acces to location not divisible by 4, an exception is generated!!!*

4. Write a short code segment (can be more than one instruction) that will branch to location **THERE** if register $10 is GREATER THAN register $11. You CANNOT use psuedo instructions, only use what is on the provided reference sheet.

   You CANNOT just use 'slt'. The opposite of LESS THAN is GREATER THAN OR EQUAL, we want to branch to THERE if $10 is GREATER THAN $11.

```
        beq  $10, $11,  skip        # if equal skip around next code
        slt  $t0, $11, $10          #  $t0 = 1 if $11 less than $10
        beq  $t0, $zero,  THERE     # if $t0 = 0, then $11 GREATER THAN $10
SKIP:   … next instruction….
```

5. Register $sp is used for the stack pointer which grows down in memory. Write a code segment that will push register $a0 onto the stack (remember to adjust the stack pointer).

```
addi  $sp, $sp, -4          # decrement stack pointer (stack grows DOWN in memory)
sw    $a0,  0($sp)          # store value
```

6. (10 pts) Write a MIPS subroutine that will find the LARGEST SIGNED number and the SMALLEST unsigned number in an array of numbers (each number is a word, ie. 32 bits). Register $a0 contains the starting address of the array, $a1 contains the number of *values* in the array.

   Return the SMALLEST UNSIGNED number in register $v0, and the LARGEST SIGNED number in $v1.

!