

YOU MUST SHOW ALL WORK TO GET ANY CREDIT! CALCULATORS ARE NOT PERMITTED.

1. In a byte addressable memory, the contents of memory location 0 is 0x0A (hex) and the contents of memory location 1 is 0x91 (hex). Assuming this is a 2's complement value in 16 bits, give the values this represents in base 10 for a: (8 points)

Little ENDIAN machine: -28406

Big ENDIAN machine: 2705

2. Convert 0x80AF (2's complement value written in hexadecimal) to the following (assume all values are in 16 bits): (12 points)

Decimal: -32593 Octal (2's complement value): 100257

2's Complement (in binary): 1000 0000 1010 1111

Signed Magnitude (in binary): 1111 1111 0101 0001

3. Add the following 8-bit 2's complement numbers and indicate whether or not overflow occurs for each one: (12 points)

11010100	10111001	01011101	00100110
10101011	11010110	00100001	01011010
-----	-----	-----	-----
01111111	10001111	01111110	10000000

Overflow? Yes

Overflow? No

Overflow? No

Overflow? Yes

4. Convert the following as indicated: (10 points)

17.625 (base 10) into base 2: 10001.101

001101101.01011 (base 2) into base 10 (decimal): 109.34375

5. List four different MIPS instructions which change the program counter as a side effect (no pseudo ops): (4 points)

JR, J, BNE, BEQ, JAL

6. List two different programming techniques for passing parameters to a function in MIPS assembly language.

Via registers (A0 through A3) or on the Stack

7. LW and SW are MIPS R2000 I-type instructions that have the following format (with size): (7 points)

OP – 6 bits	RS – 5 bits	RT – 5 bits	Address – 16 bits
-------------	-------------	-------------	-------------------

a) Why is the RS field 5 bits long?

5 bits needed to address all 32 registers (2^5)

b) The address field contains a 2's complement value for both LW and SW instructions. Why is this value a 2's complement value? (Give an example in assembly language to back up your answer).

LW \$t0, -4(\$fp) LW \$t0, 4(\$fp)

8. Complete the following program by finishing the function: char *strcat(char *s1, char *s2). Strcat() concatenates the string s2 to the end of string s1 and returns the pointer to the start of string s1. You may not alter any code given below. (Hints: .byte reserves a byte. Use SB (store byte instruction) to copy the second string SB \$src, offset(\$dest))

(30 points)

```
.data
str2: .asciiz "Bozo"
str1: .asciiz "Muller is a "
.byte 0,0,0,0
.text
.globl main
main: addiu $sp, $sp, -12
      la $t0, str1
      sw $t0, 0($sp)
      la $t0, str2
      sw $t0, 4($sp)
      sw $ra, 8($sp)
      jal strcat
      lw $ra, 8($sp)
      addiu $sp, $sp, 12
      move $a0, $v0
      li $v0, 4 #print_str
      syscall
      jr $ra
strcat:
```

strcat:

```
# get the parameters set up pointers and the
# return value (assume non-null input strings)
lw $t0, 0($sp)
lw $t1, 4($sp)
move $v0, $t0 # return value

# Find the end of string s1 byte at a time
skip: lbu $t2, 0($t0)
      addiu $t0, $t0, 1
      # if a zero in $t2, at the end
      bne $t2, $zero, skip
      addiu $t0, -1
      # adjust $t0 to point at the end of string s1

# Copy strings s2 to end of s1 byte at a time
next: lbu $t2, 0($t1)
      addiu $t1, $t1, 1
      sb $t2, 0($t0)
      addiu $t0, $t0, 1
      # we must copy the null also!
      bne $t2, $zero, next
      jr $ra
```

What is printed on the console: _____Muller is a Bozo_____

9. Debug the following MIPS program. The program is supposed to sum the integers in the array ADAT (up to the null) and print the result, but has some serious flaws. Your job is to correct the program with the minimal number of changes. (The value for print_int is 1; .word reserves a word of global space) (8 points)

```
.data
adat: .word 21,11,33,23,8,0
.text
.globl main
main: la $a0, adat
      jal sumt
      add $a0, $v0, $zero
      li $v0, 1
      syscall
      jr $ra
sumt: add $v0, $zero, $zero
top:  lw $t0, 0($a0)
      add $v0, $t0, $v0
      addiu $a0, 1
      bne $v0, $zero, top
      jr $ra
```

```
.data
adat: .word 21,11,33,23,8,0
.text
.globl main
main: la $a0, array
      move $s0, $ra
      jal sumt
      add $a0, $v0, $zero
      li $v0, 1
      syscall
      move $ra, $s0
      jr $ra
sumt: add $v0, $zero, $zero
top:  lw $t0, 0($a0)
      add $v0, $t0, $v0
      addiu $a0, 4
      bne $t0, $zero, top
      jr $ra
```