0. Sign-in Question [-1 point if not followed]: Fill out the front page correctly, and write your login at the top of each page.

1 Pointers and Struct [6pt][5min]

```
struct node{
        int value;
        struct node *next;
};

struct list{
        char foo;
        char *ptr;
        struct node *listHead;
        int *bar;
        int size;
};

struct list linklist;
```

1.1 [3pt] What is the correct C syntax to access the field value in struct node through the variable linklist?

a) linklist.listHead.value
b) linklist->listHead->value
c) linklist->listHead.value
d) linklist.listHead->value
e) none of the above

____

1.2 [3pt] On a MIPS machine with 32 bit addressing, and every word in memory must be aligned to 4 byte addressing, how many bytes does the compiler allocate for the variable linklist?

a) 14
b) 16
c) 17
d) 20
e) 24
f) none of the above

____

2 Pointers and Arrays [15pt][10min]

Below is a segment of C code:

```
char *p, *q, y=0; x[8]={0,1,2,3,4,5,6,7};
int i=0;

p = x;
q = &y;
*q = 'a';
for(i = 0; i < 6; i++){
        *p++ = *q;
}
q = p;
```

Assume the address of x is 0x400 and the address of y is 0x4008.

2.1 [3pt] What is the value of x[1] at the end of the code?

____

2.2 [3pt] What is the value of (x+1) at the end of the code?

____

2.3 [3pt] What is the value of *q at the end of the code?

____

For 2.4 and 2.5, consider the following code and assume the address of variable array is 0x10000000.

```
int array[5]={-1,0,1,2,3};

int main(){
        unsigned char *ptr1 = (((unsigned char *) array) + 3);
        unsigned char val;
        int *ptr2 = array + 3;

        val = *ptr1;

        return 0;
}
```

2.4 [3pt] What address (in hex) does ptr2 point to at the end of the program?

____

2.5 [3pt] What value does val contain at the end of the program?
a) 0 b) 2 c) 3 d) 255 e) none of the above ______________________

3 Linker and Loader [9pt][6min]

3.1 [6pt] Consider the following fragment of MIPS.

```
   la      $a0, buffer            #a
loop:
   beq     $a0, $a1, continue     #b
   lw      $t0, 0($a0)            #c
   …
   addiu  $a0, $a0, 4             #d
   jal     foo                    #e
   j       loop                   #f
continue:
   add     $t0, $0, $0
   …
```

Of the lines commented [a-f], which of these instructions are not completely determined until linking?

____

3.2 [3pt] Consider the following complete file foo.c.

```
#include<stdio.h>
int foo_bar(int *x);

int main(int argc, char *argv[]){
        int a[6], i=0, b=0;

        for(i = 0; i < 7; i++){
        a[0] = i + 42;
        }
        b = foo_bar(a);

return b;
}
```

On the command line, when the use types gcc –o foo foo.c to compile foo.c and generate the binary executables foo, which error message, if any, are generated?

a) The compiler warns that the code will run off the end of array "a".
b) The compiler issues an error, because the symbol foo_bar is not defined.
c) The linker issues an error, because the sumbol foo_bar is not defined.
d) The loader issues an error, because the symbol foo_bar is not defined.
e) The symbol foo_bar is defined and there is no error.

____

4 Number Systems [12pt][12min]

Suppose we represent signed numbers *P* and *Q* in 8 bits, using two's complement. Further suppose that the value of *P* is 0xAF.

4.1 [3pt] What is the value of *P* in base 10?

____

4.2 [3pt] What is the most negative possible value of *Q* such that the signed addition of *P* and *Q* does not overflow?

____

4.3 [3pt] What is the most positive possible value of *Q* such that the signed addition of *P* and *Q* does not overflow?

____

4.4 [3pt] Consider the following five floating point numbers.

$A = 1.0000 \times 2^{-1}$
$B = 1.0000 \times 2^{70}$
$C = 1.0000 \times 2^{70}$
$D = 1.0000 \times 2^{100}$
$E = 1.0000 \times 2^{40}$

Assume they are 32 bit single precision IEEE standard floating point numbers, and the compiled version of the following C code will be run on a IEEE floating point compliant floating point unit. Which one of the following places of parentheses will lead the result of F = A + B * C / D – E = 1.0000×2^{-1}?

a) (A+(B*C)/D)-E
b) (A+B*(C/D))-E
c) A+((B*C)/D-E)
d) A+(B*(C/D)-E)
e) all of the above because addition and multiplication are associative.

____

5 C Logical Operations [6pt][5min]

Assume N is an unsigned 32-bit integer, and given the C expression:

```
N == ( ( ( (N << 8) >> 8) & (0xFF00) ) | 0xFF0000)
```

5.1 [3pt] What is the smallest number that makes the expression true?
(Please write the 32-bit pattern down, following the bit positions indicated below the lines.)

```
_____________ -- _____________ -- _____________ -- _____________
31          24   23          16   15           8   7           0
```

5.2 [3pt] How many *different* numbers make the expression true?

____

6 MIPS Instructions [21pt][15min]

6.1 [3pt] With TAL (True Assembly Language), which of the following instruction(s) could be used to load number 0xABCD8765 into register $t0?

```
a) addi $t0, $0, 0xABCD8765
b) lui $t0, 0xABCD
   addiu $t0, $t0, 0x8765
c) lui $t0, 0xABCD
   addi $t0, $t0, 0x8765
d) lui $t0, 0xABCD
   ori $t0, $t0, 0x8765
e) none of the above
```

____

6.2 [3pt] What is the result in register $t0 after these lines of MAL (MIPS Assembly Language) are executed?

```
add $t0, $zero, $zero
ori $t0, $t0, 0xC3C3C3C3
andi $t0, $t0, 0xBBBBBBBB
ori $t0, $t0, 0x2A2A2A2A
```

a) 0x83838383
b) 0x02020202
c) 0xABABABAB
d) 0x2A2A2A2A
e) none of the above

____

6 MIPS Instruction Questions Continued…

6.3 [15pt] Find and fix the bugs in the following program that converts upper case characters in a string to lowercase characters. Assume that the address of the string is in $a0. *Please mark the error(s) on the instruction(s),* e.g. add $0, $0, $0 and write the correct instruction on the rightmost column. The left column is the C version of the correct code. Hint: There are at least 3 and at most 7 bugs here, and an instruction may have more than 1 error. There are two copies of the code for you to work. Circle the copy that you want to be graded, and cross out the one you don't want to be graded.

Useful ASCII code: 'a' = 97 'A' = 65 'Z' = 90

```
char tmp;                1 loop: lw    $t0,  0($a0)      1.____________________
char *str;               2       beq   $a0, $0, end      2.____________________
…                        3       addi  $a0, $a0, 1       3.____________________
while(tmp=*str){         4       slti  $t1, $t0, 65      4.____________________
   str++;                5       bne   $t1, $0, loop     5.____________________
   if(tmp<'A')           6       slti  $t1, $t0, 90      6.____________________
      continue;          7       bne   $t1, $0, loop     7.____________________
   if(tmp<='Z')          8       addi  $t0, $t0, 32      8.____________________
      *(str-1)+=32;      9       sw    $t0,  0($a0)      9.____________________
}                        10      j     loop              10.___________________
                         11 end:                         11.___________________
```

The following is an extra copy of question 6.3. Please circle the copy that you want to be graded, and cross out the one you don't want to be graded.

```
1 loop: lw    $t0,  0($a0)      char tmp;                1.____________________
2       beq   $a0, $0, end      char *str;               2.____________________
3       addi  $a0, $a0, 1       …                        3.____________________
4       slti  $t1, $t0, 65      while(tmp=*str){         4.____________________
5       bne   $t1, $0, loop        str++;                5.____________________
6       slti  $t1, $t0, 90         if(tmp<'A')           6.____________________
7       bne   $t1, $0, loop           continue;          7.____________________
8       addi  $t0, $t0, 32         if(tmp<='Z')          8.____________________
9       sw    $t0,  0($a0)            *(str-1)+=32;      9.____________________
10      j     loop              }                        10.___________________
11 end:                                                  11.___________________
```

7 MIPS Reverse Engineering [15pt][15min]

You have heard of hackers reverse engineer programs from binary. With the disassembler that you have written, you can use it to generate MIPS code, which you can use to write out readable C code. Try to convert the following MIPS function foo into C code.

```
foo:
     bnex $a0, L
     beqz $a1, exit1
     li $v0, 0
     jr $ra

exit1:
     li $v0, 1
     jr $ra

 L:
     addiu $sp, $sp, -16
     sw $ra, 12($sp)
     sw $a0, 8($sp)
     sw $a1, 4($sp)

     addiu $a0, $a0, -1
     jal foo
     sw $v0, 0($sp)
     lw $a0, 8($sp)
     lw $a1, 4($sp)
     addiu $a0, $a0, -1
     addiu $a1, $a1, -1
     jal foo
     lw $t0, 0($sp)
     addu $v0, $v0, $t0

     lw $ra, 12($sp)
     addiu $sp, $sp, 16
     jr $ra
```

```
int foo(int x, int y) {

  if (x==0 && y==0)
        return 1;
/* Please fill in the rest of the code below. */




























}
```

8 Self Modifying Program [6pt][7min] (This is Hard!)

This is a hard question, and you may want to save this until the end of the exam. The following code illustrates how a program can modify itself. For simplicity, there is only one instruction that is modified by the program, and assume you run this code on SPIM.

```
1     __start:
2        addiu  $t0,  $0,  0          # $t0 = 0
3        addiu  $t1,  $0,  0          # $t1 = 0
4        addiu  $t2,  $0,  2          # $t2 = 2
5     loop:
6        addiu  $t0,  $t0,  1         # $t0 = $t0 + 1
7        beq    $t0,  $t2,  next      # if ($t0 == $t2) branch to next
8        jal    loop
9     next:
10       addu   $t1,  $t1, $ra        # $t1 += $ra
11       beq    $t1,  $0,  exit       # if ($t1 == 0) branch to exit
12       lw     $t3,  0($t1)          # $t3 = mem[$t1]
13       addiu  $t3,  $t3,  1         # $t3 += 1
14       sw     $t3,  0($t1)          # store word $t3 to memory
15       j      next                  # jump to next
16    exit:
17       done                         # program finishes
```

8.1 [3pt] Please identify which line is modified by the program? ____________________

8.2 [3pt] Write out the new MIPS instruction in TAL of this line. ____________________

9 Trivia [0pt][0min]

What is the origin of the terms “big endian” and “little endian”? Hint: They were coined in 1726 A.D.

Table provided were:
MIPS operands
MIPS assembly language
Figure A.10 MIPS registers and usage convention
MIPS machine language
MIPS instruction formats

(You can find these tables in the CS61C **Patterson and Hennessy’s** *Computer Organization and Design* textbook.)