1) [10 pts] Number Representation

Convert the folowing 8-bit numbers from two's complement binary to decimal:

01110011 =

11101101 =

Convert the following decimal number to an 8-bit two's complement binary number:

-28 =

Convert from hex to 16-bit binary:

A5B7 =

Convert from single precision IEEE floating-point to decimal:

1011111110100000000000000000 =

Convert from decimal to single precision IEEE floating-point:

0.125 =

---

2) [14 pts] Short Answer. For multiple choice and true/false, circle your answer.

a) When multiplying two n-bit numbers, what is the maximum number of bits in the result?

b) When adding two n-bit numbers, what is the maximum number of bits in the result?

c) Write the MIPS instruction that you would use to set to zero all but the lowest (the least significant) bit of register $t0.

d) Give the names of two different MIPS instructions that you could use to multiply a value by 2.

e) Multiplication of two pointers is a legal operation in the C language.
[true false]

f) 2's complement representation is commonly used because it simplifies multiplication and division. [true false]

g) What is the decimal value of the most negative 2's complement 4-bit number?

h) Which of the following instructions specify an address that a linker never needs to resolve? [beq lw j]

i) In ten words or less, explain the main disadvantage of interrupts over polling.

j) In ten words or less, explain why interrupts are preferred over polling for handling I/O devices.

k) Name the MIPS instruction that you would use to transfer a word of data from a general-purpose register to an output device register.

l) What is the type of program that is used to join together multiple binary object files into an executable binary file?

m) All MIPS processor include special hardware for executing floating point instructions.
[true false]

n) What is decimal value of the exponent field for the IEEE single-precision floating point representation for infinity?

---

3) [5 points] memory allocation in C.

Consider the following structure declaration in C:

```
struct point {
float x;
float y;
};
```

Fill in the body of the function shown below. The function allocates and returns a new instance of point and initializes the x and y fields using
a and b, respectively.

```
struct point * newpoint(float a, float b)
{




}
```

---

4) [5 pts] Pointers in C:

Consider the following C function that is passed an array of integers and the array size and returns the sum of all the elements of the array:

```
int sum(int x[], int size)
{
        int a = 0; int i;

for (i = 0; i < size; i ++) a += x[i]
return(a);
}
```

Rewrite the sum function to use pointers instead of array indexing:

```
int sum (int x[], int size)
{
fill in code here

}
```

---

5) [8 Pts] Looping function In MIPS.

```
Consider the following C function:
int prod(int n, int x)
{
    int r = 0;

    while (n> 0)
    {
        r = r + x;
        n = n - 1;
    }
    return(r);
}
Rewrite the function prod using MIPS instructions. Assume that n is passed in register$a0, x in
register $ a1 and the result in register $v0.
Don't forget to use comments

Prod:   #MIPS version of C sum function
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
            jr $ra                  # return
```

6) [12 Pts] Psuedoinstruction expansion.

For each of the following pseudoinsrtuctions, produce a minimal sequence of MIPS instructions to accomplish the same thing. Remember to use only the MIPS instructions from page 2. In this problem, big refers to a specific number that requires 32 bits to represent and small, smalla, and smallb to a number that can be expressed using 16bits.

clr $t1 # t1 = 0

mv smalla($t1), smallb($t2) # MEMORY[t1 + smalla] = MEMORY[t2 + smallb]

lw $t1, big($t2) #t1 = MEMORY[t2 + big]

bge $t1, $t2, L # if (t1 >= t2) goto L

set $t1 # t1 = all ones

li $t1, big # t1 = big

7) [20 Points] Recursive funtion in MIPS

Consider the following structure declaration used for representing trees. Each node in the tree holds a pointer to a right sub-tree (or a 0 if no sub-treeis present), a pointer to a left sub-tree (or a0 if no sub-tree is present), and an integer value.

```
struct node {
        struct node right;          /* pointer to right sub-tree */
        struct node left; /* pointer to right sub-tree */
        int value;                  /* value of this node */
};
```

The function findMax is passed as pointer to a tree and returns the maximum of all the values of the nodes in the tree. It uses the function max, defined below.

```
int max(int a, int b)
{
        if (a > b) return(a) else return(b);
}

int findMax( struct node *p)
{
        if (p ==0) return (0x80000000);
        else return (max(p à  value, max(findMax(p à left), findMax(p à right)) ));
}
```

Write the equivalent MIPS assembly language function for max. Assume that the arguments a and b are passed in register $a0 and $a1, respectively, and that the return value is passed back in $v0. Don't forget comments.

```
max                                  : MIPS version of max
________________________________  #  __________________________
________________________________  #  __________________________
________________________________  #  __________________________
________________________________  #  __________________________
________________________________  #  __________________________
________________________________  #  __________________________
________________________________  #  __________________________
________________________________  #  __________________________
           jr $ra                 #
```

Write the equivalent MIPS assembly language function for findMax. The argument p is passed in register $a0, and that the return value is passed back in $v0. Assume that the order of the elements in the structure corresponds to their order in memory.

```
findMax                       # MIPS version of findMax
________________________________  #  __________________________
________________________________  #  __________________________
________________________________  #  __________________________
________________________________  #  __________________________
```

```
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________
___________________________________ # _____________________________

            jr $ra
```

---

8) [16 points] MIPS disassembly.

The following MIPS machine language program is located in memory beginning at location 0x00000000. Convert the program to MIPS assembly language.
For this problem use register numbers instead of register name; for instance use $2 instead of $v0 to the second general purpose register.

```
0001 0000 0010 0000 0000 0000 0000 0011

0000 0000 0000 0011 0001 0001 0000 0000

0010 0000 1100 0101 1111 1111 1111 1111

0000 1000 0000 0000 0000 0000 0000 0000

0000 0011 1110 0000 0000 0000 0000 0000
```

MIPS assembly language version: