CSE220 Spring 2014
Practice Final Questions

1. Convert the following binary number 0100111010011 to decimal, octal, hexadecimal, base 5.

2. Convert the decimal number (0.563_{10}) to binary for 5 positions to the right of the decimal place

3. Write -12_{10} as a 6-bit 2's complement number, 1's complement number, sign magnitude number, excess-127.

4. Consider 32-bit binary number 0111 0100 0100 0101 0110 0001 0100 1101
    a. What is the unsigned integer value? Signed integer value? IEEE single precision number?
    b. What do the ASCII characters spell in Little Endian? In Big Endian?
    c. If this value was an R-type MIPS instruction in memory, what is the value in each field? Which register is the destination register? the source register?
    d. Use a MIPS logical instruction to isolate bits 10-17 in the number (result: 0000 0000 0000 0001 0110 0000 0000 0000). Assume the value is in register $t0.
    e. Given the result of d., perform `sll $t0, $t0, 15, sra $t0, $t0, 8`. What is the value in $t0? What if the instruction was `srl` instead of `sra`?
    f. Write the single MIPS instruction to set bits 5, 10, 11 and 13 to a value of 1. Assume the value is in register $t0.
    g. Write the single MIPS instruction to divide the number by 128.
    h. Write the single MIPS instruction to multiple the number by 32.
    i. Perform the MIPS instruction `ror $t1, $t0, 6`. Assume the value is in $t0.
    j. Perform bitwise operations `and, or, nor, neg` of the value with hexadecimal value 0x6F470E4E. Write the MIPS instructions assuming the following statement type $s0 = $t0 OPERATION $t1

5. As a function, what are the MIPS register conventions that you should follow? As a leaf function, what conventions must you follow? Who's responsibility is it to maintain $t register values? The $fp? $s registers? $at?

6. Translate the following statement into MIPS assembly: A = (B - C - 30 + D – E + F)/8. Assume that variable A-F are in registers $s0-$s5 respectively. Also assume that you cannot overwrite any variable values (except A) since they are used later. Use the minimal number of additional registers (other than $s0-$s5).

7. What is the output of the following program?

```
#include <stdio.h>
int main()
{
 int *ptr1, *ptr2, i = 10, j = 20;

 ptr1 = &i;
 ptr2 = &j;

 ptr2 = ptr1;
 *ptr1 = *ptr1 + *ptr2;
```

```
 *ptr2 = 2*(*ptr2);
 printf("Val = %d\n", *ptr1 + *ptr2);
 return 0;
}
```

8. Rewrite the code in problem 7 in MIPS.
9. What are the values of `arr` elements in the following program?

```
#include <stdio.h>
int main()
{
 int *ptr, arr[5] = {20};

 for(ptr = arr+1; ptr <= arr+4; ptr++)
 *ptr = *(ptr-1) + *(ptr+1) + 1;
 return 0;
}
```

10. Rewrite the code in Problem 9 in MIPS.
11. What is the output of the following program?

```
#include <stdio.h>
#include <string.h>
int main()
{
 char str[10] = "test";
 printf("%d %s\n", *strcpy(str, "n")**strcpy(str+2, "xt"), str);
 return 0;
}
```

12. Rewrite the following code in MIPS.

```
#include <stdio.h>

int a = 4;

int main()
{
      if(a == 0)
            return 0;
      else
      {
            printf("%d ", a--);
            main();
      }
      return 0;
}
```

13. Consider the following static variables are allocated in memory beginning at address 5000.

```
      int *P;
      double A = 3.1416;
      int B = 20;
      int *C = &B;
```

```
int D = *C;
char E[25] = "Hello CSE220!!"
int F[5] = {12, 66, 96, 32, 49};
```

Determine the numerical values for the following expressions: &B, C, D, E, &E[7], &(F[1]).

14. Write the .data section of the variable declarations from Problem 13 in MIPS.

15. Write the MIPS code to assign the value of element [A][B][C] in integer array G to the value 99. The base address of G is in $s0. A, B and C are in $t0, $t1, and $t2 respectively. Modify only registers $t8 and $t9. Assume G is declared as `int G[8][5][16].`

16. Create a new pseudo instruction for MARS called decrement, which decreases the value in the register by 1. (Ex: `dec $t0, 1`) Define how the assembler should assemble and translate this pseudo instruction into actual instructions. If the instruction was to be added as an actual instruction (not pseudo), what instruction format should this instruction have and why?

17. Consider the code of length.asm (attached to the end of this document). Assume that the code is assembled and the address of main is byte 100 (word 25). The .data segment begins at byte 500. Create the symbol table after the first pass of the assembler. What is the immediate value of the beqz instruction (in decimal). What is the binary value in the 26-bit field of the `j nextCh` instruction.

The following 4 questions appeared on previous Final exams.

18. Translate the following C code statement (not loop) into MIPS assembly:

    ```
    A[i] = (B[i] -3) + B[i+4] – (2*B[i-1])
    ```

    Assume that A is a word array and B is a byte array. Both arrays are stored in memory and i is within the array bounds.

    Assume the following register assignments: Base address of A: $s1, Base address of B: $s2, i: $t0.

19. Add a for loop for i=3 to i=10 around the statement.

20. Consider the following recursive C program for calculating the greatest common denominator of two numbers.

```
int GCD(int m, int n)
{
      int result = 0;
      if(m == n)
            result = m;
      else if (m > n)
            result = GCD(m-n, n);
      else
            result = GCD(m, n-m);
      return result;
}
int main()
{
      int i, j;
```

```
        printf("Enter the first number:");
        scanf("%d", &i);
        printf("Enter the second number:");
        scanf("%d", &j);
        printf("The GCD is : %d\n", GCD(i,j));
        return 0;
}
```

(a) Define the MIPS .data section for the above program.

(b) Implement the recursive GCD function in MIPS. You may use pseudo instruction and MARS formats without penalty, but points will be taken off for inefficient code.

21. Consider the following globally defined C struct and pointer:

```
struct entry{
        char * name;
        int x;
        struct entry *next;
};
struct entry *AllData;
```

(a) Write a function in C, `printData()`, which uses a pointer to traverse the AllData list and print the name and x value (one entry per line).

(b) Write a function in C, `addtoEnd(char* name, int x)`, which creates a new struct instance at the end of the AllData list.

22. Write MIPS assembly code for the following Java program fragment, which computes the Ackermann function. You must follow all MIPS function call standards. You may use pseudo instructions, but points will be taken off for inefficient code. (Hint: Don't forget to use the stack.)

```
int Ackermann(int x, int y)
{
        int result;
        if(x == 0)
                result = y + 1;
        else if(y == 0)
                result = Ackerman(x-1,1)
        else
                result = Ackerman(x-1, Ackerman(x, y-1));
        return result;
}
```

23. The following bitcount function counts the number of 0-bits in its integer argument. Write the MIPS program that implements this function.

```
int zeroCount(unsigned x)
{
        int b;
        for (b = 0; x != 0; x >>= 1) {
                b += x & 01;
        }
        return 32-b;
}
```

```
##
## length.asm - prints out the length of character
## string "str"
##
##  t0 - holds each byte from string in turn
##  t1 - contains count of characters
##  t2 - points to the string
##

#####################################################################
#
#           text segment
#
#####################################################################

    .text
    .globl main
main:           # execution starts here
    la $t2,str  # t2 points to the strint
    li $t1,0    # t1 holds the count
nextCh: lb $t0,($t2)    # get a bye from string
    beqz $t0,strEnd # zero means end of string
    add $t1,$t1,1   # increment count
    add $t2,$t2,1   # move pointer one character
    j nextCh    # go round the loop again

strEnd: la $a0,ans  # system call to print
    li $v0,4    # out a message
    syscall

    move $a0,$t1    # system call to print
    li $v0,1    # out the lenghth worked out
    syscall

    la $a0,end1 # system call to print
    li $v0,4    # out a newline
    syscall

    la $v0,10
    syscall     # au revoir...

#####################################################################
#
#           data segment
#
#####################################################################

    .data
str:    .asciiz "hello world"
ans:    .asciiz "Length is "
end1:   .asciiz "\n"

##
## end of file length.asm
```