ECE 550
Homework 3
Due October 19

For this homework you will be answering 2 "pencil and paper" questions, and doing one programming questions. Your submission should be a tar or zip file containing two other files:

- A pdf or plain text file with your answers to questions 1–2.
- The finished `strsort.c` and `strsort.s` file for question 3.

Please remember that **no Word Documents or rtf files** will be accepted.

Shengxin Qian
Xinquan Liu

Question 1: Stack Frame

Suppose you have a function which you want to write in MIPS assembly. This function writes to the following registers: `$s0`, `$s1`, `$s2`, `$t0`, `$t1`, `$t2`, `$a0`, `$a1`, `$v0`. Of these, `$s0`, `$s1`, and `$t1` are live across calls to other functions.

1. Write the MIPS assembly code to setup your stack frame, and save the registers you need to save at the start of function.

```
addiu $sp, $sp, -32
sw    $fp, 0($sp)
sw    $ra, 4($sp)
sw    $s0, 8($sp)
sw    $s1, 12($sp)
sw    $s2, 16($sp)
addiu $fp, $sp, 28
```

2. Write the MIPS assembly code to restore the registers you need to and clean up the stack frame at the end of this function.

```
lw    $s2, 16($sp)
lw    $s1, 12($sp)
lw    $s0, 8($sp)
lw    $ra, 4($sp)
lw    $fp, 0($sp)
addiu $sp, $sp, 32
```

3. Describe where else you need to save and restore registers, and state which ones you need to save and restore.

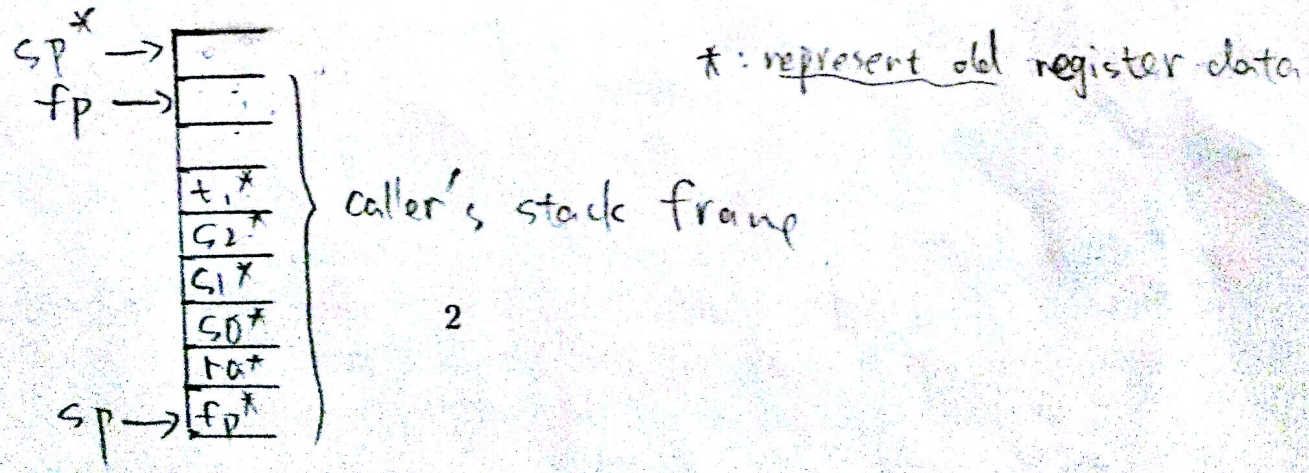
when call other function:

```
sw $t1, 20($sp)
```

when return from other function

```
lw $t1, 20($sp)
```

4. Draw a diagram of your stack frame (as you have set it up). Include the stack pointer and frame pointer, and show what register is saved in each slot. Finally, indicate where the caller's stack frame exists.



Question 2: Assembly bits

For each of the following tasks (a) write a MIPS assembly instruction to perform the task, and (b) convert the MIPS assembly you wrote to its numeric encoding.

- Add `$t1 + $a1` and store the result in register `$s2`

R-type

op	rs	rt	rd	shamt	funct
000000	01001	00101	10010	00000	100000

add $s2, $t1, $a1

0x0125 9020

- Load an unsigned byte from the address `$sp + 16` into register `$t0`

I-type

op	rs	rt	imme
100100	11101	01000	0000 0000 0001 0000

lbu $t0, 16($sp)

0x93A80010

- Branch if `$a0` is equal to `$s0` to label `endIf` (which is 40 instructions away from the current instruction).

beq $a0, $s0, endIf

000100 00100 10000 0000 0000 0010 0111

0x1090 0027

1) the offset in MIPS
2) the binary code

- Call the function `factorial`, whose address is `0x1234000`

jal factorial

000011 0001 0010 0011 0100 0000 0000 00

0x0C48D000