1. (6 points) A string is palindrome if it reads the same backwards and forwards. Examples of palindromes include the strings: "x", "madam", "wow" and "avid diva". Write a procedure, ***palindrome*** in MIPS assembly language that checks if a string is a palindrome. The procedure should take a single argument that is a pointer to a null-terminated string in register $a0. The ***palindrome*** procedure should return a one in $v0 if the string is palindrome and 0 in $v0 otherwise. Comment your code. Assume that the string has as least one character. Do NOT use pseudo-instructions.

```
        addi  $t1, $zero, 0
loop1:  add   $t2, $a0, $t1
        lb    $t0, 0($t2)
        beq   $t0, $zero, length
        addi  $t1, $t1, 1
        j loop1
length: addi  $t3, $zero, 0
        srl   $t5, $t1, 1
loop2:  add   $t6, $a0, $t3
        lb    $t7, 0($t6)
        sub   $t8, $t1, $t3
        add   $t8, $t8, $a0
        lb    $t9, 0($t8)
        bne   $t7, $t9, done
        addi  $t3, $t3, 1
        beq   $t5, $t3, complete
        j loop2
done:   add $v0, $zero, $zero
        j exit
complete: addi $v0, $zero, 1
exit:   jr $ra
```

addi $t1, $t1, -1

Finding length (3)

(3)

2. (6 points) In the embedded market, where cost is crucial, processors sometimes implement floating point only in software. We are interested in two implementations of a computer, one with and one without special floating-point hardware.
Consider a program, P, with the following mix of operations:

Floating-point multiply	10%
Floating-point add	15%
Floating-point divide	5%
Integer instructions	70%

Computer MFP (computer with floating point) has floating-point hardware and can therefore implement the floating-point operations directly. It requires the following number of clock cycles for each instruction class:

Floating-point multiply	6
Floating-point add	4
Floating-point divide	20
Integer instructions	2

Computer MNFP (computer with no floating point) has no floating-point hardware and so must emulate the floating-point operations using integer instructions.
The integer instructions all take 2 clock cycles. The number of integer instructions needed to implement each of the floating-point operations is as follows:

Floating-point multiply	30
Floating-point add	20
Floating-point divide	50

Both computers have a clock rate of 1000 MHz. Find the mega instruction per second ratings for both computers.

CPI for MFP = $0.1 \times 6 + 0.15 \times 4 + 0.05 \times 20 + 0.7 \times 2 = 3.6$ (1)

CPI for MNFP = 2 (1)

Mega instruction per second for MFP = $\frac{1000}{3.6} = 277.7$ (0.5)

" " " " MNFP = $\frac{1000}{2} = 500$ (0.5)

If the computer MFP needs 300 million instructions for this program, how many integer instructions does the computer MNFP require for the same program?

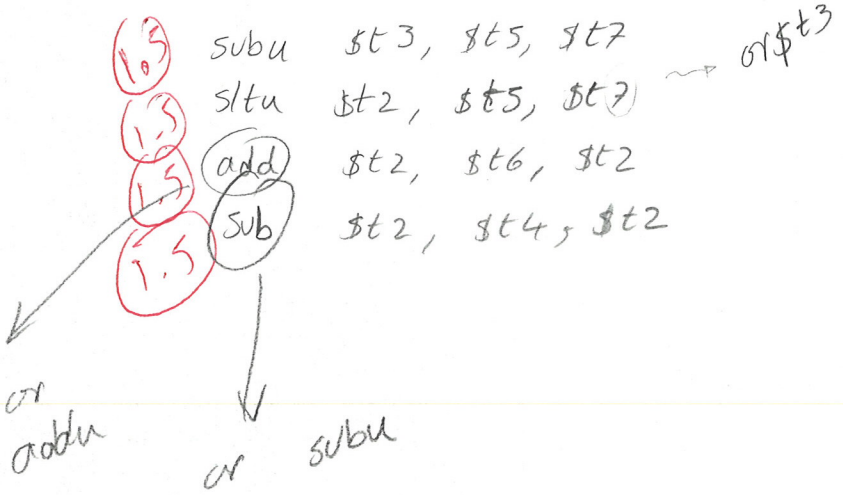
What is the execution time (in seconds) for the program P to run on MFP and MNFP?

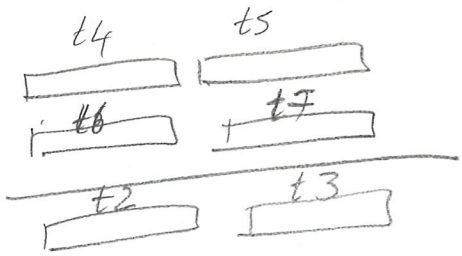
	MFP		MNFP
FP Mult.	$10\% \times 300 = 30$	→	900
" add.	$15\% \times 300 = 45$	→	900
FP Div.	$5\% \times 300 = 15$	→	750
Integer	$70\% \times 300 = 210$	→	210
	300		2760 (1)

Execution time for MFP = $\frac{300}{277} = 1.08$ sec (1)

" " " MNFP = $\frac{2760}{500} = 5.52$ sec (1)

3. (6 points) Find the shortest sequence of MIPS instructions to perform 64-bit integer subtraction. Assume that one 64-bit, two's complement integer is in registers $t4 and $t5 and another is in registers $t6 and $t7. The result is to be placed in registers $t2 and $t3. In this example, the most significant word of the 64-bit integer is found in the even-numbered registers, and the least significant word is found in the odd-numbered registers. (Hint: It can be done in four instructions.)

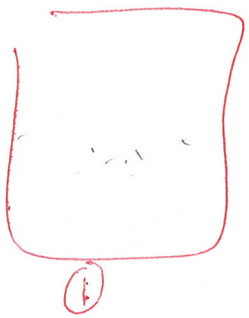


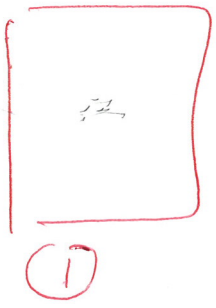


4. (6 points) Briefly explain the motivation for using Dynamically Linked Libraries and how DLL works. Draw two diagrams: one showing steps for the first time a call is made to the DLL routine and the second one showing steps in subsequent calls.

Motivation: • In static linking library becomes part of the code.
① This prevent using newer versions of the library
• Static linking loads the whole library even
① if all of the library is not used.

DLL: library routines are not linked and loaded until the program is run.





explanation
②

5. (6 points) In combinatorial mathematics, the Catalan numbers form a sequence of natural numbers that occur in various counting problems. They are named for the Belgian mathematician Eugène Charles Catalan (1814–1894). The definition of Catalan number is:
Cat(1)=1
Otherwise;

$$Cat(n) = \frac{2(2n-1)}{n+1} Cat(n-1)$$

The C code for Cat(n) would be:

```
int Cat(n){
        if (n==1) {
          return 1;
        }else{
          return 2*(2*n-1)/(n+1)Cat(n-1)
        }
        }
```

Write a recursive MIPS procedure to calculate Cat(n). Assume that n is in $a0 and your result should be placed in $v0.

```
Cat :   addi  $sp, $sp, -8
        sw    $ra, 4($sp)        # back up ra
        sw    $a0, 0($sp)        # back up a0
        addi  $t1, $zero, 1
        beq   $a0, $t1, Ret1     # if n=1 goto Ret1
        addi  $t1, $a0, 1        # n+1
        sll   $t2, $a0, 1        # 2n
        addi  $t2, $t2, -1       # 2n-1
        sll   $t2, $t2, 1        # 2(2n-1)
        mult  $t2, $v0           # 2(2n-1)/(n+1)
        mflo  $t2
        div   $t2, $t1
        mflo  $v0
        addi  $sp, $sp, 8
        jr    $ra
Ret1 :  addi  $v0, $zero, 1
        addi  $sp, $sp, 8
        jr    $ra
```

```
addi $a0, $a0, -1
jal cat
lw $a0, 0($sp)
lw $ra, 4($sp)
```