COMSATS University Islamabad, Lahore Campus

Assignment 2 – FALL 2024

Course Title:	Computer Organization & Architecture				Course Code:	CPE 343	Credit Hours: 4(3,1)
Course Instructor:	Dr. Muhammad Naeem Awais				Program Name:	BCE	
Semester:	5th	Batch:	FA21	Section:	A, B	Date Given:	8th October, 2024
Submission Date:	**15th October, 2024**				**Maximum Marks:**		**30**
Name:					**Registration Number:**		

Important Instructions / Guidelines:

- Draw neat schematics wherever needed
- Do your own work, PLAGARISM will be graded as ZERO
- **No late submission**.

Question 1: **[CLO3-PLO2-C3]** **[10 Marks]**

For the given piece of C code, ***produce*** the equivalent MIPS assembly code.

```
For (i = 20; i >= 0; i = i - 1)
        W[i+1] = X[i-1] + s*Y[i];
```

While generating the assembly code, assume that `W`, `X` and `Y` are arrays and their base addresses are in registers $s0 to $s2. Whereas `s` is a 32-bit number that corresponds to $t0 and `i` is an array index that corresponds to $t1.

Solution:-

```
addi $t0, $zero, s          # s(32-bit)initialized in t0
addi $t1, $zero, 20         # initial index value
addi $t2, $zero, -1         # final index value

Loop: beq   $t1, $t2, Exit  # check whether i=<50
      sll   $t3, $t1, 2     # index multiplied by 4
      add   $t4, $t3, $s0   # base address of W in s0
      add   $t5, $t3, $s1   # base address of X in s1
      add   $t6, $t3, $s2   # base address of Y in s2
      lw    $t7, -4($t5)    # load at effective address of X
      lw    $t8, 0($t6)     # load at effective address of Y
      mul   $t9, $t8, $t0   # multiply Y[i] with s
      add   $s3, $t7, $t9   # addition of X[i-1] + s*Y[i]
      sw    $s3, 4($t4)     # store at effective address of W
      addi  $t1, $t1,-1     # increment in index value
      j Loop
Exit:
```

Question 2: **[CLO3-PLO2-C3]** **[10 Marks]**

For the given piece of C code, ***produce*** the equivalent MIPS assembly code.

```
While   (s*Y[i] != X[j])

          X[j] = X[i] + 100;

          i += 1;

          j += 1;
```

While generating the assembly code, assume that `X` and `Y` are arrays and their base addresses are in registers $s0 to $s1. Whereas `s` is a 32-bit number that corresponds to $t0 and `i` and j are array indices that corresponds to $t1 and $t2 respectively.

Solution:-

```
addi $t0, $zero, s            # s(32-bit)initialized in t0
addi $t1, $zero, 0            # initial index value of i
addi $t2, $zero, 0            # initial index value of j

Loop:     sll       $t3, $t1, 2    # index i multiplied by 4
          sll       $t4, $t2, 2    # index j multiplied by 4

          add       $t5, $t4, $s0  # effective address of X[j]
          lw        $t6, 0($t5)
          add       $t7, $t3, $s1  # effective address of Y[i]
          lw        $t8, 0($t7)
          mul       $t9, $t8, $t0
          beq       $t7, $t9, Exit


          add       $s2, $t3, $s0  # effective address of X[i]
          lw        $s3, 0($s2)
          addi      $s4, $s3, 100
          add       $s5, $t4, $s1  # effective address of Y[j]
          sw        $s4, 0($s5)


          addi      $t1, $t1, 1
          addi      $t2, $t2, 1
          j       Loop
Exit:
```

Question 3: **[CLO3-PLO2-C3]** **[10 Marks]**

For the given piece of C code, ***produce*** the equivalent MIPS assembly code using **STACK**:

```
int exampleprocedure (int g, int h, int i)
     {int f = 0;

     while (g!=h)
     {f = f*i*X[h];h--;}

     return f;}
```

While translating the code, assume that g, h, i correspond to parameter registers $a0 to $a2 and f corresponds to $s0 whereas X is an array and its base address is stored in registers $s1. Assume array X is accessed in descending order and has total 10 elements.

Solution:-

```
Exampleprocedure:

addi $sp, $sp, -32
sw   $t0, 28($sp)
sw   $t1, 24($sp)
sw   $t2, 20($sp)
sw   $t3, 16($sp)
sw   $t4, 12($sp)
sw   $t5, 8($sp)
sw   $s0, 4($sp)
sw   $s1, 0($sp)


add  $t0, $a0, $zero     # variable g
add  $t1, $a1, $zero     # variable h
add  $t2, $a2, $zero     # variable i

add  $s0, $zero, $zero   # variable f
addi $t1, $zero, 10      # initialization of index h



while:
     sll  $t3, $t1, 2
     beq  $t0, $t1, Exit
     mul  $s0, $s0, $t2
     add  $t4, $s1, $t3
     lw   $t5,0($t4)
     mul  $s0, $s0, $t5
     addi $t1, $t1, -1
j    while

Exit:
     add  $v0, $s0, $zero
```

```
        lw   $s1, 0($sp)
        lw   $s0, 4($sp)
        lw   $t5, 8($sp)
        lw   $t4, 12($sp)
        lw   $t3, 16($sp)
        lw   $t2, 20($sp)
        lw   $t1, 24($sp)
        lw   $t0, 28($sp)
        addi $sp, $sp, 32

jr   $ra
```