CS F342 Computer Architecture

Semester 1 – 2023-24

Lab Sheet 11

Goals for the Lab: We build up on prior labs and Exploring sorting techniques using MIPS Exercise 1: Write a program to implement C bubble sort program given below in MIPS.

C program code:

```
int main() {
        int Sz = 10;
        int List[10] = {17, 5, 92, 87,41, 10, 23, 55, 72, 36} ;
        int Stop, // $s3: upper limit for pass
        Curr, // $s0: index of current value in comparison
        Next, // $s1: index of successor to current value
        Temp; // $s2: temp storage for swap
        for (Stop = Sz-1; Stop > 0; Stop) {
                for (Curr = 0; Curr < Stop; Curr++) {
                        Next = Curr + 1;
                        if ( List[Curr] > List[Next] ) {
                                Temp = List[Curr];
                                List[Curr] = List[Next];
                                List[Next] = Temp;
                        }
                }
        }
        printf("Sorted list in ascending order:\n");
        for (Curr = 0; Curr < Stop; Curr++)
                printf("%d\n", List[Curr]);
}
```

Hint *::* To convert Curr to offset you can use sll $t4, $t2, 2 or similar where $t2 is Curr, $t4 is offset from starting address of buffer abd shift of 2 implies multiplying by 4.

Partial assembly code: (Highlighted part is complete)

```
        .data
        list: .word 17, 5, 92, 87,41, 10, 23, 55, 72, 36
        space: .asciiz " "
        .text
        main:
                li $s7,10               #size of the list(sz)
                addi $s3,$s7,-1         # $s3 = Stop = sz-1
                #Write the loop, swap code here

        exit:                           #print the array
                la $t0,list
                li $t2,0                #as a counter while printing the list
        print:
                lw $a0,($t0)            #load current word in $a0
                li $v0,1
                syscall                 #print the current word
                la $a0,space
                li $v0,4
                syscall                 #print space in b/w words
                addi $t0,$t0,4          #point to next word
                addi $t2,$t2,1          #counter++
                blt $t2,$s7,print
```

```
          li $v0,10          #exit MIPS progam
          syscall
```

Exercise 2: Write a program to implement above program but store floating point numbers instead of integer.

Hint: Use commands swc1, lwc1, c.le.s, bc1f, bc1t
Comparison of FP values sets a code in a special register and Branch instructions jump depending on the value of the code:
c.le.s $f2, $f4 #if $f2 <= $f4 then code = 1 else code =
0 bc1f label   #if code == 0 then jump to label bc1t label #
if code == 1 then jump to label

Exercise 3: write a program to implement C Insertion sort program given below in MIPS.

C program code:

```
int main() {
        int n = 5;
        int array[5] = { 5, 3, 4, 2, 1 };
        int c = 0;
        int d = 0;
        int t = 0;
        for (c = 1 ; c <= n - 1; c++) {
                d = c;
                while (d > 0 && array[d] < array[d - 1]) {
                        t = array[d];
                        array[d] = array[d - 1];
                        array[d - 1] = t;
                        d--;
                }
        }
        for (c = 0; c <= n - 1; c++) {
                printf("%d\n", array[c]);
        }
        return 0;
}
```

Partial assembly code:

```
.data
array: .word 0 : 1000       # an array of word, for storing values.
size: .word 5               # actual count of the elements in the array.

sort_prep:
        la $t0, array       # load array to $t0.
        lw $t1, size        # load array size to $t1.
        li $t2, 1           # loop runner, starting from 1.

sort_xloop:
        la $t0, array       # load array to $t0.
        bge $t2, $t1, sort_xloop_end # while (t2 < $t1).
        move $t3, $t2       # copy $t2 to $t3.

sort_iloop:
        la $t0, array       # load array to $t0.
        mul $t5, $t3, 4     # multiply $t3 with 4, and store in $t5
        add $t0, $t0, $t5   # add the array address with $t5, which is the index multiplied with 4.
        ble $t3, $zero, sort_iloop_end # while (t3 > 0).
        lw $t7, 0($t0) # load array[$t3] to $t7.
        lw $t6, -4($t0) # load array[$t3 - 1] to $t6.
        bge $t7, $t6, sort_iloop_end # while (array[$t3] < array[$t3 - 1]).
        lw $t4, 0($t0)
        sw $t6, 0($t0)
        sw $t4, -4($t0)
        subi $t3, $t3, 1
        j sort_iloop # jump back to the beginning of the sort_iloop.

sort_iloop_end:
        addi $t2, $t2, 1 # increment loop runner by 1.
        j sort_xloop # jump back to the beginning of the sort_xloop.

sort_xloop_end :
        li $v0, 4 # 4 = print_string syscall.
        la $a0, sorted_array_string # load sorted_array_string to argument register $a0.
        syscall # issue a system call.
        li $v0, 4 # 4 = print_string syscall.
        la $a0, line # load line to argument register $a0.
        syscall # issue a system call.

        jal print # call print routine.
```