Lab 04 – Worksheet

Name: Aqsa Muneer, Maleeha Nasir Khan	ID: 10527, 09991	Section: T1

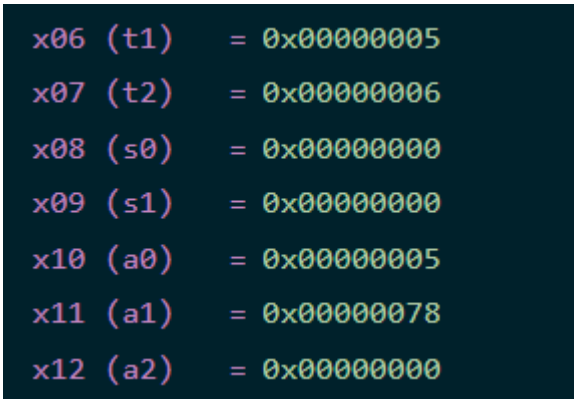
Note: Assumptions and logics should be explained separately in tasks after the task results.

Task 1

Provide appropriately commented codes.

```
.text
.globl main
fact:
    li x10, 0 # i = 0
    li x7, 1 #starting value of f
    li x6, 5 #eg number for loop end of loop
    li x11, 1 #product/factorial

loop1:
    beq x10, x6, exit
    mul x11, x11, x7 #f=i*f
    addi x7, x7, 1
    addi x10, x10, 1 #increment i
    j loop1

exit:
    j exit
```

**Add snip of relevant Registers / Memory locations where you think results are obtained. Make sure the irrelevant area of snip is cropped. Make sure the irrelevant area of snip is cropped.*



Comments

Algorithm initializes a product register (x11) to 1, then iterates from 0 to 5, multiplying the product by an incrementing value (x7) on each iteration. The loop continues until the counter reaches the target value, at which point the factorial result is stored in x11 and the program exits.

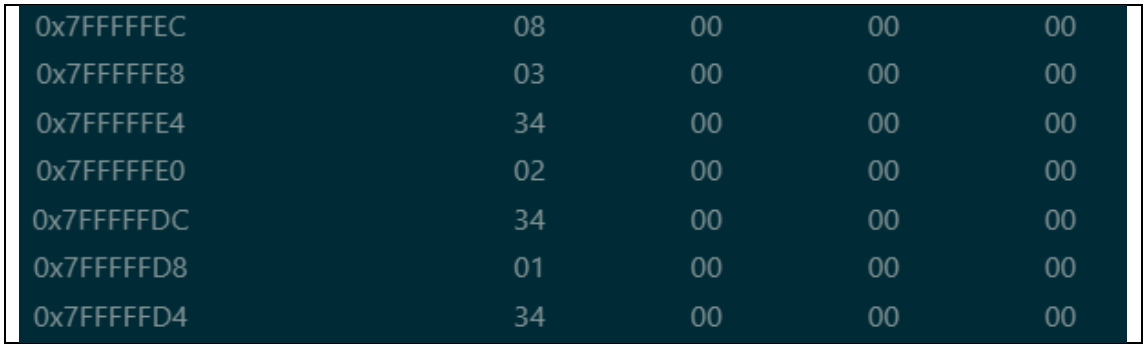
Task 2

Provide appropriately commented codes

```
.text
.globl

main:
    addi x10, x0, 3
    jal x1, ntri

    addi x11, x10, 0
    li x10, 1
    ecall
    j exit

ntri:
    addi sp, sp, -8

    sw x1, 4(sp)
    sw x10, 0(sp)

    #base case check
    addi x5, x10, -1 #x5 = num - 1
    blt x10, x0, ntri_base

ntri_recursive:
    addi x10, x10, -1
    jal x1, ntri #recursive call
    addi x6, x10, 0

    lw x10, 0(sp)
    lw x1, 4(sp)
    addi sp, sp, 8

    add x10, x10, x6
    jalr x0, 0(x1)
```

```
ntri_base:
    addi x10, x0, 0
    addi sp, sp, 4
    jalr x0, 0(x1)

exit:
    j exit
```

Add screenshot of your results



0x7FFFFFEC	08	00	00	00
0x7FFFFFE8	03	00	00	00
0x7FFFFFE4	34	00	00	00
0x7FFFFFE0	02	00	00	00
0x7FFFFFDC	34	00	00	00
0x7FFFFFD8	01	00	00	00
0x7FFFFFD4	34	00	00	00

Comments

The main program passes 3 as input to the ntri function via x10. The function uses the stack to save the return address and current parameter, then checks the base case where n < 0 (returns 0). For the recursive case, it decrements n, recursively calls ntri, and adds the returned result to the current n value. After each recursive call returns, the function restores saved values from the stack and returns the accumulated sum via x10 to the caller using jalr.

Task 3

Provide appropriately commented codes

```
.text
.globl
```

```
main:
    li x10, 0x200
    #populate the array
    li x5, 8
    sw x5, 0(x10)
    li x5, 20
    sw x5, 4(x10)
    li x5, 15
    sw x5, 8(x10)
    li x5, 3
    sw x5, 12(x10)
    li x5, 9
    sw x5, 16(x10)

    jal x1, bubble_sort #call bubble sort

bubble_sort:
    #allocatian of stack
    addi sp, sp, -8
    sw x8,8(sp)
    sw x9, 4(sp)

    #bubble sort Loqic
    li x7, 0 #i
    li x9, 4

outer_loop:
    bge x7, x9, exit
    li x8, 0 #j=0
    li x6, 0 #swapped = false
    sub x9, x9, x7 #Limit of j = n-i-1

inner_loop:
    bge x8, x9, check_flag

    # Load a[j] and a[j+1]
    slli x5, x8, 2
    add x5, x10, x5
    lw x20, 0(x5)
    lw x21, 4(x5)

    ble x28, x21, no_swap

    # swap
    sw x21, 0(x5)
    sw x20, 4(x5)
    li x6, 1
```

```
no_swap:
    addi x8, x8, 1
    j inner_loop

check_flag:
    beq x6, x0, exit
    addi x7, x7, 1
    j outer_loop

exit:
    j exit
```

Add screenshots of your results

Address	+0	+1	+2	+3
0x00000218	00	00	00	00
0x00000214	00	00	00	00
0x00000210	09	00	00	00
0x0000020C	03	00	00	00
0x00000208	0F	00	00	00
0x00000204	14	00	00	00
0x00000200	08	00	00	00

Before sorting

Address	+0	+1	+2	+3
0x00000218	00	00	00	00
0x00000214	00	00	00	00
0x00000210	09	00	00	00
0x0000020C	03	00	00	00
0x00000208	0F	00	00	00
0x00000204	14	00	00	00
0x00000200	08	00	00	00

After sorting

Comments:

The algorithm uses nested loops where the outer loop tracks passes (i) and the inner loop compares adjacent elements (j). For each pair of adjacent elements, the code loads a[j] and a[j+1] using bit-shift indexing (multiply by 4 for byte offset), then swaps them if a[j] > a[j+1] to achieve ascending order. A swapped flag tracks whether any swaps occurred in the current pass; if no swaps happened, the array is sorted and the function exits. Stack allocation preserves x8 and x9 registers, and the array size decreases by one each outer loop iteration to optimize subsequent passes.

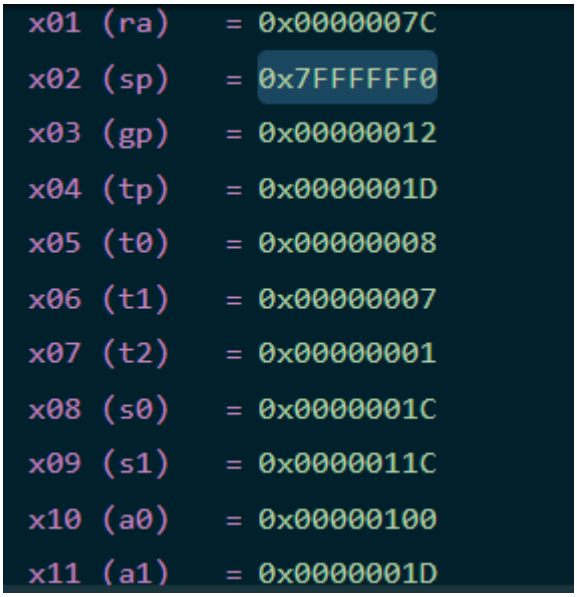
Task 4

Provide appropriately commented codes

```
.text
.globl main
main:
    li x10,0x100
    li x6,7
    addi sp,sp,-4
    sw x1,0(sp)
    li x3,2
    li x4,1
    sw x3,0(x10)
    sw x4,4(x10)
    beq x6,x0, done_L0
    li x7,1
    beq x6,x7,done_L1
    li x5,2
    jal x1,loop
    j finish

done_L0:
    addi x11,x3,0
    j finish

done_L1:
    addi x11,x4,0
    j finish

loop:
    add x11, x3, x4
    add x8, x5, x5
```

```
    add x8, x8, x8
    add x9, x10, x8
    sw x11, 0(x9)
    addi x3, x4, 0
    addi x4, x11, 0
    addi x5, x5, 1
    ble x5, x6, loop
    jalr x0, 0(x1)

finish:
    lw x1,0(sp)
    addi sp,sp,4

exit:
    j exit
```

Add screenshots of your results



Address	+0	+1	+2	+3
0x00000130	00	00	00	00
0x0000012C	00	00	00	00
0x00000128	00	00	00	00
0x00000124	00	00	00	00
0x00000120	00	00	00	00
0x0000011C	1D	00	00	00
0x00000118	12	00	00	00
0x00000114	0B	00	00	00
0x00000110	07	00	00	00
0x0000010C	04	00	00	00
0x00000108	03	00	00	00
0x00000104	01	00	00	00
0x00000100	02	00	00	00

Comments:

Implemented Fibonacci sequence that computes the nth Fibonacci number where n=7. The program initializes an array at address 0x100 with the first two Fibonacci numbers (2 and 1), then handles edge cases: if n=0 returns the first value, if n=1 returns the second value. For n≥2, the loop iterates from index 2 to n, computing each Fibonacci number as the sum of the previous two values (x11 = x3 + x4), storing results in the array at calculated offsets, and updating the running values. The offset calculation uses bit-shifting (multiply by 4 via left-shift and addition) to access 4-byte integer array elements. The final result is stored in x11 and the function exits after restoring the return address from the stack.

Lab 04 – Nest Procedures and Sorting
Assessment Rubrics

Points Distribution

Task No.	LR2 Code	LR 5 Results	AR 7 Report Submission & Github
Task 1	/10	/05	/10 & /10
Task 2	/10	/10	
Task 3	/10	/10	
Task 4	/15	/10	
Total Points	/100 Points		
CLO Mapped	CLO 2		

For description of different levels of the mapped rubrics, please refer the provided Lab Evaluation Assessment Rubrics and Affective Domain Assessment Rubrics.

#	Assessment Elements	Level 1: Unsatisfactory Points 0-1	Level 2: Developing Points 2	Level 3:Good Points 3	Level 4:Exemplary Points 4
LR2	**Program/Code / Simulation Model/ Network Model**	Program/code/simulation model/network model does not implement the required functionality and has several errors. The student is not able to utilize even the basic tools of the software.	Program/code/simulation model/network model has some errors and does not produce completely accurate results. Student has limited command on the basic tools of the software.	Program/code/simulation model/network model gives correct output but not efficiently implemented or implemented by computationally complex routine.	Program/code/simulation /network model is efficiently implemented and gives correct output. Student has full command on the basic tools of the software.
LR5	**Results & Plots**	Figures/ graphs / tables are not developed or are poorly constructed with erroneous results. Titles, captions, units are not mentioned. Data is presented in an obscure manner.	Figures, graphs and tables are drawn but contain errors. Titles, captions, units are not accurate. Data presentation is not too clear.	All figures, graphs, tables are correctly drawn but contain minor errors or some of the details are missing.	Figures / graphs / tables are correctly drawn and appropriate titles/captions and proper units are mentioned. Data presentation is systematic.
AR9	**Report**	All the in-lab tasks are not included in report and / or the report is submitted too late.	Most of the tasks are included in report but are not well explained. All the necessary figures / plots are not included.	Good summary of most the in-lab tasks is included in report. The work is supported by figures and plots with explanations. The report is submitted timely.	Detailed summary of the in-lab tasks is provided. All tasks are included and explained well. Data is presented clearly including all the necessary figures, plots and tables.

			Report is submitted after due date.		