/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Name: Daniel Ackuaku \*/

/\* Course: Engineering 304L \*/

/\* Lab: Lab Number 3 \*/

/\* Date: 21st February 2019. \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* The ".text" assemlber directive indicates the beginning of the code section of the program \*/

.text

/\*TEMPLATE: factorial function \*/

/\* The ".global factorial" assembler directive exports the \*/

/\* "factorial" label as an external symbol, so that C-Code can call it \*/

.global factorial

factorial: /\* Starting location of the function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FunctionTemplate\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Pseudo-code for this function is as follows

/\* pop the top elemment from the stack

/\* decrement the stack pointer (sp)

/\* if n = 1,

/\* return 1

/\* else stroe the value of n

/\* subtract 1 from n

/\* repeat above steps

/\* multiply n \* (n-1)

/\*

/\* store each value of n in memory using a stack

/\* increment the sp by 8

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*TEMPLATE: rename the function labels as desired below \*/

/\*TEMPLATE: write the assembly function below \*/

FACTORIAL\_INIT:

/\* include any register pushes here \*/

addi sp, sp, -0x8

stw ra, 4(sp)

stw r16, 0(sp)

FACTORIAL\_EXEC:

/\*TEMPLATE: include main execution of the function here \*/

beq r4, r0, BASE\_CASE

mov r16, r4

subi r4, r4, 1

call factorial

mul r2, r2, r16

br FOUND\_N

BASE\_CASE:

/\*TEMPLATE: include any register pops here \*/

movi r2, 1

/\* Note that there should only be one "ret" instruction in your function \*/

FOUND\_N:

ldw r16, 0(sp)

ldw ra, 4(sp)

addi sp, sp, 0x8

ret /\* instruction to return to the calling function \*/

.end /\* the assembler throws away all text after this line \*/

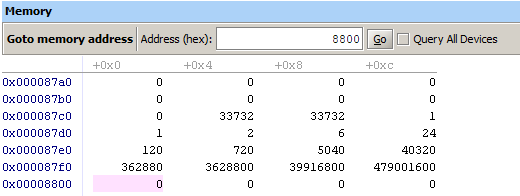


Figure . showing the results of the assembly code

Table A table containing the uses of the various registers in the program.

|  |  |
| --- | --- |
| Registers | Uses |
| r2 | Returns the value of factorial(n) |
| r3 | The location of the last data entry + 4 |
| r4 | Stores the current value of n when the recursive portion of the function is being carried out. |
| r7 | The location of the last data entry + 4 |
| r17 | The location of the current computation would be stored |
| r18 | The initial value entered into the factorial function |

Table A table showing the location and size of variables in the c code.

|  |  |  |
| --- | --- | --- |
| Variable | Register / Memory | Size in bytes |
| int Fact [13] | 087cc -0087f0 | (4\*13) = 52 bytes |
| int n | r16 | N/A |