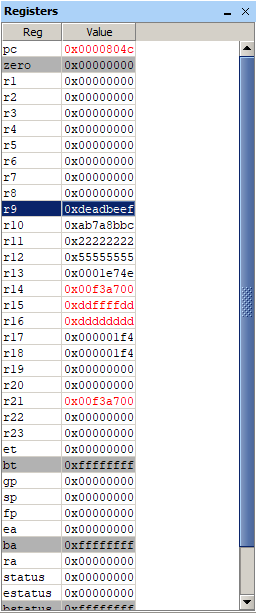
**Engr 304 SW Lab 1 Name: Daniel Ackuaku**

**Section: B**

|  |  |  |  |
| --- | --- | --- | --- |
| **Constant or Variable** | **Register Assigned to hold the value** | **Final Hex Value at the end of execution** | **Final Decimal Value at the end of execution** |
| A | r9 | 0xdeadbeef | 3735928559 |
| B | r10 | 0xab7a8bbc | 2876935100 |
| C | r11 | 0x22222222 | 572662306 |
| X | r12 | 0x55555555 | 1431655765 |
| Y | r13 | 0x0001e74e | 124750 |
| Z | r14 | 0x00f3a700 | 15968000 |
| W | r15 | 0xddffffdd | 3724541917 |
| not C | r16 | 0xdddddddd | 3722304989 |

Screen capture (snip) of your register usage:



Assembly code listing: (see notes in the Formatting Software Printouts document on formatting this properly)

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

/\* Name: Daniel Ackuaku \*/

/\* Course: Engineering 304L \*/

/\* Lab: 1 \*/

/\* Date: 2/7/2019 \*/

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

/\*TEMPLATE: fill in the title block above \*/

/\* The ".include nios\_macros.s" assembler directive includes NIOS macros for use in the program \*/

.include "nios\_macros.s"

/\* The word-aligned address of the reset vector, value is taken from the cpu configuration in SOPC builder. \*/

/\*TEMPLATE: replace 0x00 with the appropriate address \*/

.equ RESET\_VECTOR, 0x00

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

.text

/\* The ".org RESET\_VECTOR" assembler directive places the main routine at the reset address \*/

.org RESET\_VECTOR

/\* The ".global \_start" assembler directive exports the "\_start" label as an external symbol \*/

.global \_start

/\* The "\_start" label identifies the program start location for the debugger \*/

\_start:

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

/\*\*\*\* MAIN \*\*\*\*/

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

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

/\* Initialize variables (sum), initialize registers, etc

Begin LOOP

Read value from array

If value=0, exit the loop, else

Add value to sum variable

Increment loop index (or pointer address)

return to "Begin LOOP"

\*/

/\*\*\*\* init \*\*\*\*/

MAIN\_PROG\_INIT:

movia r9, 0xDEADBEEF /\* load the hexadecimal number into the r9 register \*/

movia r10, 0xAB7A8BBC /\* load the number 2876935100 into the r10 register \*/

movia r11, 0x22222222 /\* load the number 572662306 into the r11 register \*/

movi r17, 0x1F4 /\* load the number 500 into the r17 register \*/

movi r18, 0x1 /\* load the number 1 into the r18 register \*/

/\*\*\*\* run \*\*\*\*/

SUM\_OF\_Y:

add r13, r13, r18 /\* add r18 to the value in r13 (1 + Y ) \*/

addi r18, r18, 1 /\* increment r18 by 1 \*/

bne r18, r17, SUM\_OF\_Y /\* loop back to the top if r18 is not 500 \*/

/\*\*\*\* run \*\*\*\*/

SUM\_OF\_X:

sub r12, r9, r10

add r12, r12, r11

/\*\*\*\* run \*\*\*\*/

Z\_XOR:

srai r20, r14, 3 /\* shift r14 3 bits to the right and store in r20 \*/

slli r21, r13, 7 /\* shift r14 7 bits to the left and store in r21 \*/

xor r14, r20, r21 /\* r14 = r20 xor r21 \*/

/\*\*\*\* run \*\*\*\*/

W:

movi r22, 0 /\* load the hexadecimal number into the r22 register \*/

nor r16 , r22, r11 /\* create a not using thee nor of the zero register and r11 storing the result in r16 \*/

or r15, r14, r16 /\* r15 = r14 xor r21 \*/

/\*\*\*\* destroy \*\*\*\*/

MAIN\_PROG\_END:

br MAIN\_PROG\_END /\* infinite loop to keep program from going into the weeds \*/

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

/\*\*\*\* DATA \*\*\*\*/

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

/\* The ".data" directive identifies the section of the program that defines global variables \*/

.data

/\* The ".end" assembler directive indicates the end of the program and

all following lines are discarded \*/

.end