**LAB REPORT**

**IT3280E– 152049– Assembly Language and Computer Architecture Lab**

**Lab 03: Jump and Branch Instructions**

# **Assignment 1:**

*Create a new project to implement the code in Home Assignment 1. Initialize for i and j variable. Compile and upload to the simulator. Run this program step by step, observe the changing of memory and the content of registers at each step.*

*A screenshot of a computer

Description automatically generated*

* Content of Registers**:**

+ Step 1: initialize *i* = 5



+ Step 2: initialize *j* = 3



+ Step 3: initialize *x* = 0



+ Step 4: initialize *y* = 10



+ Step 5: initialize *z* = 0



+ Step 6: set *t0* = 1 since *j* < *i* (in this case 3 < 5)



+ Step 7: compare *i* and *j* => since *j* < *i*, we jump to else

+ Step 8: *y* = *y*-1 = 9



+ Step 9: *z* = 2\**z* = 0



* The memory remained unchanged since this program mainly use registers
* Key Observations:

+ When the program compares *i* and *j* using the **slt** instruction (*s1* and *s2*), the result is stored in *t0*. If *t0* = 1, the program will jump to the *else* part.

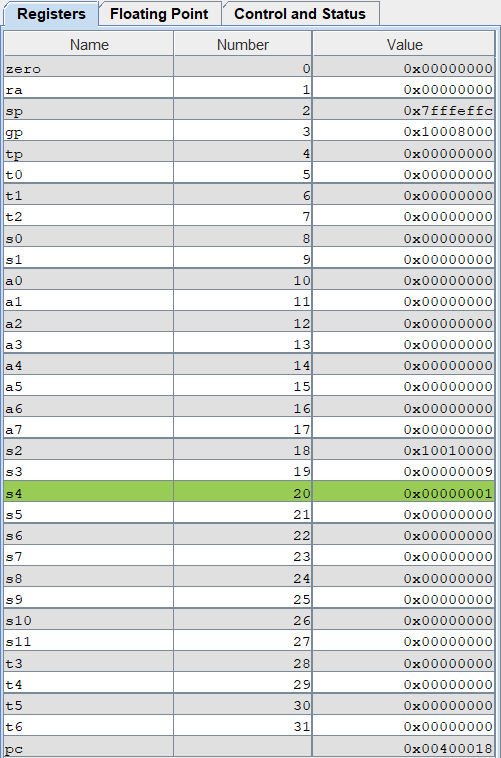
+ After the *then* or *else* part, the program skips to the *endif* label.

**Assignment 2:**

*Create a new project implementing the code in Home Assignment 2. Initialize for i, n, step, sum variables and array A. Compile and upload to the simulator. Run this program step by step, observe the changing of memory and the content of registers by each step. Try to test with some more cases (change the value of variables).*

**Case 1: A screenshot of a computer

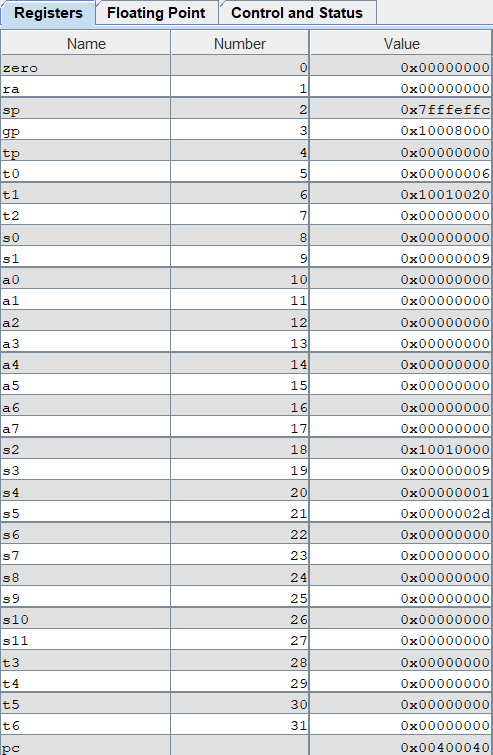
Description automatically generated**



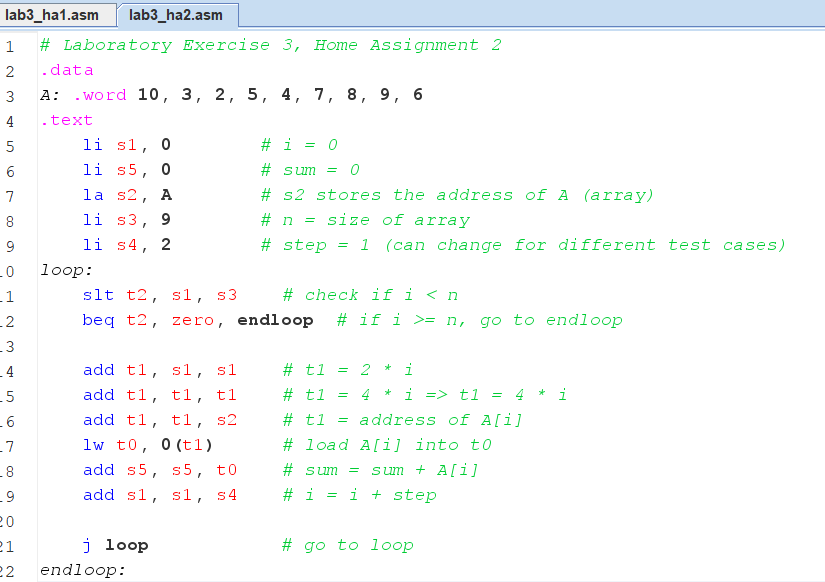
* Content of Registers:

+ Initialization:

* *i* = *s1* = 0
* sum = *s5* = 0
* *s2* = address of A = 0x10010000
* *n* = *s3* = size of array = 0x00000009
* step = *s4* = 1 = 0x00000001

+ After loop:

* *s1* (loop index i) = 0x00000009 = 9 in decimal
* *s5* (sum) = 0x0000002D = 45 in decimal
* *s3* (n) = 0x00000009 = 9 in decimal
* *s2* (address of the array A) = 0x10010000 (the base address of A in memory)
* *s4* (step) = 0x00000001 = 1 in decimal
* The memory remained unchanged since this program mainly use registers

**Case 2: **

* *increment by steps of 2 instead of 1 and change the values in A (1 -> 10)*
* A table with numbers and a green line

  Description automatically generatedContent of Registers:

+ Initialization:

* *i* = *s1* = 0
* sum = *s5* = 0
* *s2* = address of A = 0x10010000
* *n* = *s3* = size of array = 0x00000009
* step = *s4* = 2 = 0x00000002

A table of numbers and symbols

Description automatically generated+ After loop:

* *s1* (loop index i) = 0x00000009 = 9 in decimal
* *s5* (sum) = 0x0000002D = 45 in decimal
* *s3* (n) = 0x00000009 = 9 in decimal
* *s2* (address of the array A) = 0x10010000 (the base address of A in memory)
* *s4* (step) = 0x00000001 = 1 in decimal
* The memory remained unchanged since this program mainly use registers

# **Assignment 3:**

*Create a new project implementing the code in Home Assignment 3. Compile and upload to the simulator. Run this program step by step; observe the changing of memory and the content of registers by each step. Change the value of test variable and run this program some times to check all cases.*

A screenshot of a computer

Description automatically generated

* A table with numbers and a green line

  Description automatically generatedContent of Registers:

+ Initialization:

* *s0* = address of test
* *s1* = value of test = 0
* *t0* = 0
* *t1* = 1
* *t2* = 2
* Program compares *s1* to *t0, t1, t2* to decide which block of code to run

A table of numbers and letters

Description automatically generated+ Case 0 (test = 0):

* **addi** *s2, s2, 1*: This increments the value of register *s2* by 1.
* After this, the program jumps to the continue label to end execution.

A screenshot of a table

Description automatically generated+ Case 1 (test = 1):

* **sub** *s2, s2, t1*: This subtracts 1 from *s2* (since *t1* is 1).
* Then, it jumps to the **continue** label to finish.

A white sheet with numbers and lines

Description automatically generated with medium confidence+ Case 2 (test = 2):

* **add** *s3, s3, s3*: This doubles the value of *s3* (multiplies it by 2).
* After this, it jumps to **continue**.
* *The memory remained unchanged in this programm. Only the registers changed*

# **Assignment 4:**

1. i < j:

A screenshot of a computer program

Description automatically generated

1. i >= j:

A screenshot of a computer

Description automatically generated

1. i + j <= 0:

A screenshot of a computer

Description automatically generated

1. i + j > m + n (với m và n được lưu trong các thanh ghi khác)

A screenshot of a computer

Description automatically generated

# **Assignment 5:**

1. i <= n

A screenshot of a computer

Description automatically generated

1. sum >= 0

A screenshot of a computer

Description automatically generated

1. A[i] != 0

A screenshot of a computer program

Description automatically generated

# **Assignment 6:**

A screenshot of a computer

Description automatically generatedA white background with green text

Description automatically generated