**IT3280E Computer Architecture Lab**

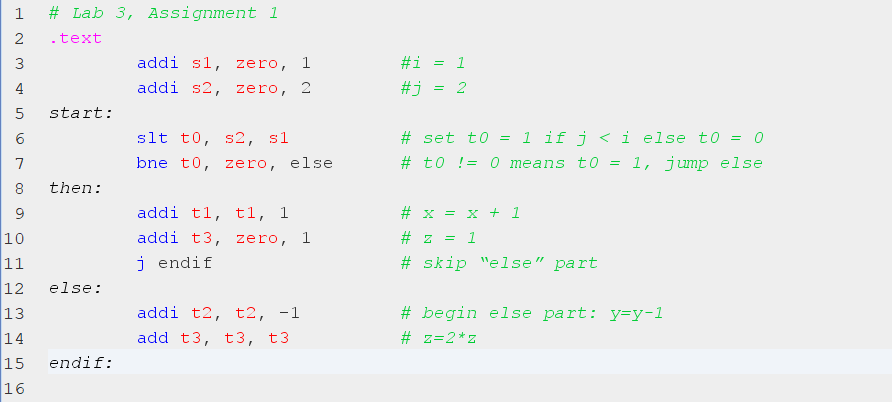
**Nguyễn Đức Anh**

**MSSV: 20235890**

**Lab 3. Jump and Branch Instructions**

**Assignment 1**

Source code:



On this assignment, we have introduced the condition branch instructions and uncondition branch instructions and extended instruction (j label), jump to label (target address) without condition.

* **beq rs1, rs2, label**:  
  Branch if equal.  
  Condition: if (rs1 == rs2) PC = BTA  
  Meaning: The program counter (PC) jumps to the branch target address (BTA) if rs1 equals rs2.
* **bne rs1, rs2, label**:  
  Branch if not equal.  
  Condition: if (rs1 ≠ rs2) PC = BTA  
  Meaning: The PC jumps to BTA if rs1 is not equal to rs2.
* **blt rs1, rs2, label**:  
  Branch if less than.  
  Condition: if (rs1 < rs2) PC = BTA  
  Meaning: The PC jumps to BTA if rs1 is less than rs2.
* **bge rs1, rs2, label**:  
  Branch if greater than or equal to.  
  Condition: if (rs1 ≥ rs2) PC = BTA  
  Meaning: The PC jumps to BTA if rs1 is greater than or equal to rs2.
* **bltu rs1, rs2, label**:  
  Branch if less than (unsigned).  
  Condition: if (rs1 < rs2) PC = BTA  
  Meaning: The PC jumps to BTA if rs1 is less than rs2 in an unsigned comparison.
* **bgeu rs1, rs2, label**:  
  Branch if greater than or equal to (unsigned).  
  Condition: if (rs1 ≥ rs2) PC = BTA  
  Meaning: The PC jumps to BTA if rs1 is greater than or equal to rs2 in an unsigned comparison.
* **jalr rd, rs1, imm**:  
  Jump and link register.  
  Action: PC = rs1 + SignExt(imm), rd = PC + 4  
  Meaning: The PC is set to the value in rs1 plus the sign-extended immediate value. The return address is saved in rd.
* **jal rd, label**:  
  Jump and link.  
  Action: PC = JTA, rd = PC + 4  
  Meaning: The PC is set to the jump target address (JTA), and the return address is saved in rd.

**A screenshot of a computer

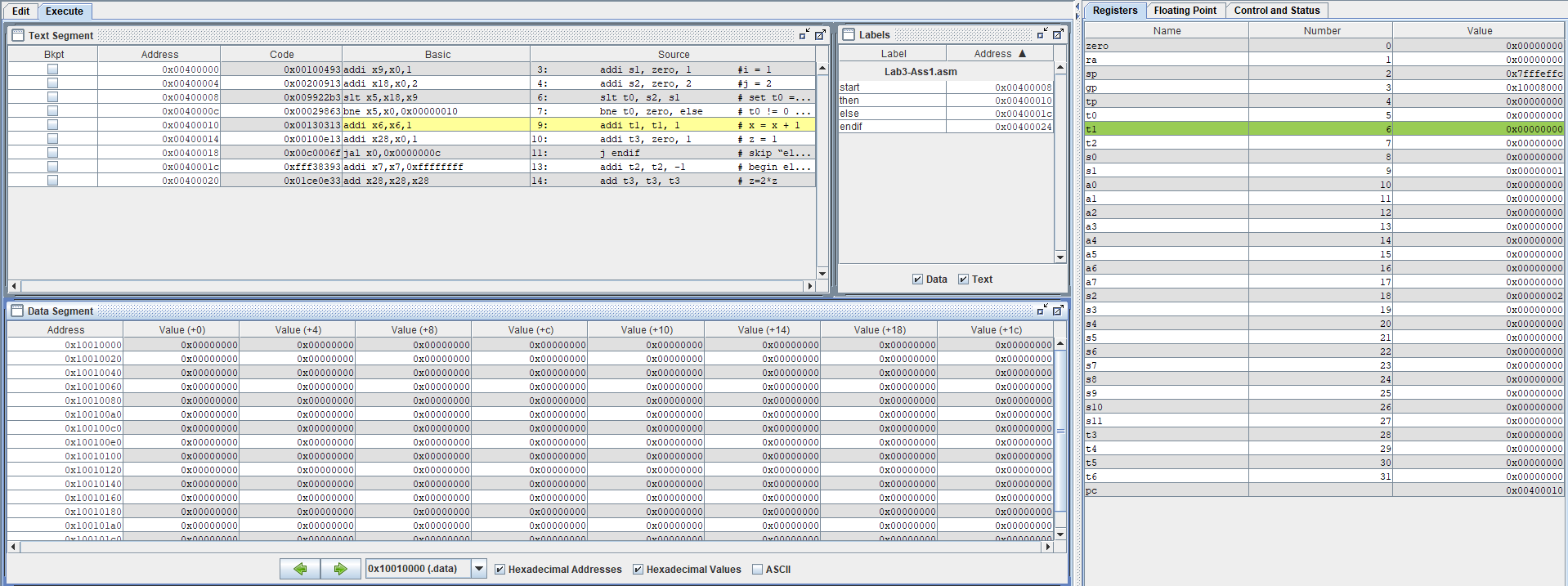
Description automatically generated**

* First, we assign s1 = 1 (i = 1), s2 = 2 (j = 2).
* On the Register window, the value of s1 if 0x00000001, s2 is 0x00000002 and the PC (program counter) is 0x00400008 as the address of the line 6.

**A screenshot of a computer

Description automatically generated**

* In the instruction slt, compare i and j if j < i, t0 = 1 else t0 = 0.
* In this case, i = 1, j = 2 so j > i, t0 = 0. The value of t0 is 0x00000000 in the Register window.



* Because t0 = 0 so when the bne instruction check if the value of t0 = 1, it goes to “else” statement or it goes to then statement. So in this case, it goes to then statement.

**A screenshot of a computer

Description automatically generated**

* After run line 9 and 10 of the “Then” statement, In the Register window, the value of t1 is 0x00000001 and t3 is 0x00000001 refer to value we want to assign in the source code. The PC value is 0x00400018.

A screenshot of a computer

Description automatically generated

* After run the “j endif”, it means that we jump into endif without condition, the value of PC is 0x00400024 and we end the program.

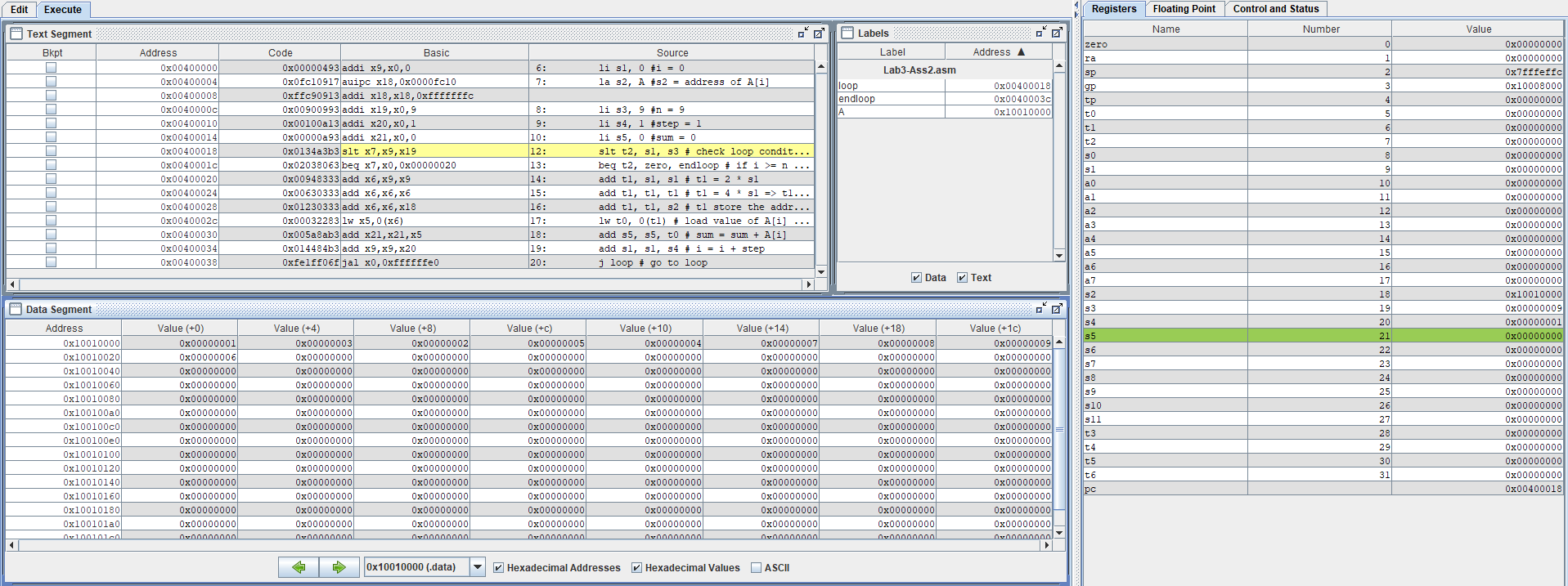
**Assignment 2**

Source code:

A screenshot of a computer program

Description automatically generated

On this assignment, we will use loop to make the sum of elements of array A.



* Like in the source code comment, we assign s1 = 0 (i = 0), s2 store the address of A in this case, the first element is A[0] so s2 value is the address of A[0], s3 = 9 is the number of elements, s4 = 1 is step, s5 = 0 is the sum of the elements of array A.
* After run from line 6 to line 10 to assign all the elements, in the Register window, the value of s1 is 0x00000000, s2 is 0x10010000, s3 is 0x00000009, s4 is 0x00000001, s5 is 0x00000000 and the PC is 0x00400018 is the address of line 12.
* In the Data Segment window, we see that refers to the address of A, the value of each elements are represented on to the address increases by 4 byte.

A screenshot of a computer

Description automatically generated

* We run the slt in line 12 to check loop condition if i < n so t = 1 result on Register window t2 value is 0x00000001.
* After run line 13, because t2 = 1 != 0 so the program begin to run the add to calculate the offset t1 = 4 \* i and lw from A[i] to t0.

A screenshot of a computer

Description automatically generated

* After run line 14 to line 17, because i = 0 so the offset 4 \* i = 0, so the address store on t1 is 0x10010000 refers to the address of A[0] in the Data Segment, going with that is the value of A[0] store in t0 is 0x00000001 in Register window.

A screenshot of a computer

Description automatically generated

* Run line 18 and 19, on the Register window, we can see that the value of s1 (as step counting) and s5 (as sum of elements) updated to 0x00000001, the PC value is 0x00400038 as the address of line 20 is jump back to the loop.

A screenshot of a computer

Description automatically generated

* After line 20, the loop begins again until satisfy the loop condition is that i >= n.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

* After 9 loops, the value of s1 goes to 0x00000009 >= n = 9 so the value of t0 changes to 0x00000000 and jump to the endloop statement.
* After another run, the PC value changes to 0x0040003c, it means the end of the program.

A screenshot of a computer

Description automatically generated

* The result of sum we have is the value of s5 is 0x0000002d in hexadecimal and 45 in decimal.

**Assignment 3**

Source code:

A screenshot of a computer

Description automatically generated

On this assignment, we use another type of branch instruction is switch/case.

A screenshot of a computer

Description automatically generated

* First, we load the address of test to s0, in the Register window the value of s0 is changed to 0x10010000 as the address of test in Labels window.

A screenshot of a computer

Description automatically generated

* Second, we load the word value of s0 (as test) to s1. Because we declare the value of test on the .data is 0 so the value of s1 is 0x00000000.

A screenshot of a computer

Description automatically generated

* Third, we load immediate the value of t0, t1 and t2 to 0, 1, 2 and in the Register window the value 0x00000000, 0x00000001, 0x00000002 refer to the value of t0, t1 and t2.
* In this situation, the PC value is 0x00400018 means that it’s ready to run the line 10.
* From the line 10 to 12, we make the comparison between s1 and t0 or t1 or t2. If it satisfies one of three conditions it will jump to the case that refers with that condition.

A screenshot of a computer

Description automatically generated

* In this case, we know that the value of s1 is 0x00000000, so it’s equivalent to the value of t0 is also 0x00000000. So that it jumps to the case\_0 refers to the line 10 conditional checking.
* The PC value changes to 0x00400028 refers to the line 15 in the case\_0.

A screenshot of a computer

Description automatically generated

* The value of s2 changes from 0x00000000 to 0x00000001 refers to the addi instruction of case\_0. The program continues to the j continue and stop the program from doing next case.

A screenshot of a computer

Description automatically generated

We change the value of test to 1 to test others case if it works or not.

A screenshot of a computer

Description automatically generated

* We load the address of test to s0 and load the word value of test to s1.
* On the Register window, the value of s0 is 0x10010000 and s1 is 0x00000001.

A screenshot of a computer

Description automatically generated

* Run addi to add the value of t0 = 0, t1 = 1 and t2 = 2, we begin to check the condition.
* The PC value is 0x00400018.

A screenshot of a computer

Description automatically generated

* After run line 13, Because the value of s1 is 0x00000001 != the value of t0 is 0x00000000 so it skips the line 13. The PC value changes to 0x0000001c refers to the address of the line 14.

A screenshot of a computer

Description automatically generated

* After run line 14, The PC value changes to 0x00400030 refers to the line 21 in the statement of case\_1. This occurs because s1 = 0x00000001 equal to t1 = 0x00000001.

A screenshot of a computer

Description automatically generated

* We run the instruction of case\_1 is subtracting the value s2 by 1 and the value of s2 in the Register window changes from 0x00000000 to 0xFFFFFFFF (means 0 – 1= - 1).

A screenshot of a computer

Description automatically generated

* We run the j continue instruction to jump into continue statement and end the program. The PC value is 0x00400040 and the program end.

A screenshot of a computer

Description automatically generated

We change the value of test to 2 to test others case if it works or not.

A screenshot of a computer

Description automatically generated

* We load the address of test to s0 and load the word value of test to s1.
* On the Register window, the value of s0 is 0x10010000 and s1 is 0x00000002.

A screenshot of a computer

Description automatically generated

* Run addi to add the value of t0 = 0, t1 = 1 and t2 = 2, we begin to check the condition.
* The PC value is 0x00400018.

A screenshot of a computer

Description automatically generated

* After run line 13 and 14, Because the value of s1 is 0x00000002 != the value of t0 is 0x00000000 and the value of t1 is 0x00000001 so it skips the line 13 and 14. The PC value changes to 0x00000020 refers to the address of the line 15.

A screenshot of a computer

Description automatically generated

* After run line 15, The PC value changes to 0x00400038 refers to the line 24 in the statement of case\_2. This occurs because s1 = 0x00000002 equal to t2 = 0x00000002.

A screenshot of a computer

Description automatically generated

* We run the instruction of case\_2 is adding the value s3 with s3 and the value of s3 in the Register window is 0x00000000 because s3 is 0x00000000, when we add 2 times it means 0 \* 2 = 0.

A screenshot of a computer

Description automatically generated

* We run the j continue instruction to jump into continue statement and end the program. The PC value is 0x00400040 and the program end.

A screenshot of a computer

Description automatically generated

* We change the value of test to 2 to test others case if it works or not.

A screenshot of a computer

Description automatically generated

* We load the address of test to s0 and load the word value of test to s1.
* On the Register window, the value of s0 is 0x10010000 and s1 is 0x00000003.

A screenshot of a computer

Description automatically generated

* Run addi to add the value of t0 = 0, t1 = 1 and t2 = 2, we begin to check the condition.
* The PC value is 0x00400018.

A screenshot of a computer

Description automatically generated

* After run line 13, 14 and 15, Because the value of s1 is 0x00000003 != the value of t0 is 0x00000000, t1 is 0x00000001 and t2 is 0x00000002 so it skips the line 13 , 14 and 15. The PC value changes to 0x00000024 refers to the address of the line 16.

A screenshot of a computer

Description automatically generated

* We run the j default instruction to jump into default statement and end the program. The PC value is 0x00400040 and the program end.

**Assignment 4**

Part A:

Source code:

A screenshot of a computer

Description automatically generated

* In part a, we modify by changing the position of i and j to implement of statement if i < j.

Part B:

Source code:

A screenshot of a computer

Description automatically generated

* In part b, we modify by changing bne to beq, so that it’s reversed of the assignment 2 by the statement i >= j.

Part C:

Source code:

A screenshot of a white board with green text

Description automatically generated

* In part c, we assign t0 = i + j, we modify by using extended instruction blez (branch if less than or equal to zero) to implement the statement i + j <= 0.

Syntax: **blez t1, label** Branch if Less than or Equal to Zero: Branch to statement at label if t1 <= 0.

Part D:

Source code:

A screenshot of a computer

Description automatically generated

* In part d, we modify by declaring variables s3 (as m) = 3 and s4 (as n) = 4, assigning t0 = i + j and t1 = m + n. In this case, t0 = 3, t1 = 7, because t1 > t0 so t2 = 0 and jump to else statement.

**Assignment 5**

Part A:

Source code:

A screenshot of a computer

Description automatically generated

* In part a, we modify by using a extended instruction ble (branch less or equal) to implement the condition i <= n. The condition will be unsatisfied if i > n. In this case, if i = 9, the program still loop until i = 10, the program will jump to endloop.

Syntax: **ble t1, t2, label** Branch if Less or Equal : Branch to statement at label if t1 <= t2

Part B:

Source code:

A screenshot of a computer code

Description automatically generated

* In part b, we modify by using a extended instruction bgez (branch if greater than or equal to Zero) to implement the condition sum >= 0. The condition will be unsatisfied if the first element of array add to the sum is negative number or the final result of the sum is a negative number, the program will jump to endloop.

Syntax: **bgez t1, label** Branch if Greater than or Equal to Zero: Branch to statement at label if t1 >= 0

Part C:

Source code:

A screenshot of a computer program

Description automatically generated

* In part c, we modify by using a extended instruction beqz (branch if equal to Zero) to implement the condition A[i] != 0. The condition will be unsatisfied if one of the elements of the array equal to zero, the program will jump to endloop.
* Syntax: **beqz t1, label** Branch if Equal to Zero: Branch to statement at label if t1 == 0