Họ và tên : Đỗ Mạnh Phương

MSSV : 20225660

LABORATORY EXCERCISE 3: Load/ Store, Jump & 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.

**\*Trường hợp 1: i = j = 4**

A screenshot of a computer

Description automatically generated

* Bảng trạng thái của các thanh ghi:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Trạng thái | Nội dung thanh ghi | | | | | | |
| $s1 | $s2 | $t0 | $t1 | $t2 | $t3 | Pc |
| Ban đầu | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00400000 |
| Chạy lệnh addi 1 | 0x00000004 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00400004 |
| Chạy lệnh addi 2 | 0x00000004 | 0x00000004 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00400008 |
| Chạy lệnh slt | 0x00000004 | 0x00000004 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x0040000c |
| Chạy lệnh bne | 0x00000004 | 0x00000004 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00400010 |
| Chạy lệnh addi 1 của start | 0x00000004 | 0x00000004 | 0x00000000 | 0x00000001 | 0x00000000 | 0x00000000 | 0x00400014 |
| Chạy lệnh addi 2 của start | 0x00000004 | 0x00000004 | 0x00000000 | 0x00000001 | 0x00000000 | 0x00000001 | 0x00400018 |
| Kết thúc lệnh endif | 0x00000004 | 0x00000004 | 0x00000000 | 0x00000001 | 0x00000000 | 0x00000001 | 0x00400024 |

* Nhận xét:

Ta nhận thấy khi giá trị của i ứng với thanh ghi $s1 = (0x00000004) và giá trị của j ứng với thanh ghi $s2 = (0x00000004) thì giá trị của thanh ghi $t0 sẽ luôn bằng 0 xuyên suốt chương trình. Có hiện tượng này xảy ra vì ở câu lệnh slt, điều kiện j<i không đúng nên thanh ghi $t0 có giá trị = 0. Khi đến câu lệnh bne (**B**ranch If **N**ot **E**qual), giá trị của thanh ghi $t0 bằng với giá trị của $zero nên không thực hiện rẽ nhánh đến else, vì vậy, chương trình sẽ tiếp tục thực hiện phần code ở start. Gán giá trị cho thanh ghi $t1 bằng 1 (0x00000001) và gán giá trị cho thanh ghi $t3 bằng 1 (0x00000001). Ngoài ra, giá trị của thanh ghi pc cũng tăng 24 so với giá trị ban đầu, tương ứng với 8 câu lệnh mà ta đã thực hiện ở chương trình.

**\*Trường hợp 2: j = 1 , i = 2 (j<i)**

A screenshot of a computer program

Description automatically generated

* Bảng trạng thái:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Trạng thái | Nội dung thanh ghi | | | | | | |
| $s1 | $s2 | $t0 | $t1 | $t2 | $t3 | pc |
| Ban đầu | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00400000 |
| Chạy lệnh addi 1 | 0x00000002 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00400004 |
| Chạy lệnh addi 2 | 0x00000002 | 0x00000001 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00400008 |
| Chạy lệnh slt | 0x00000002 | 0x00000001 | 0x00000001 | 0x00000000 | 0x00000000 | 0x00000000 | 0x0040000c |
| Chạy lệnh bne | 0x00000002 | 0x00000001 | 0x00000001 | 0x00000000 | 0x00000000 | 0x00000000 | 0x0040001c |
| Chạy lệnh addi của else | 0x00000002 | 0x00000001 | 0x00000001 | 0x00000000 | 0xffffffff | 0x00000000 | 0x00400020 |
| Chạy lệnh add của else | 0x00000002 | 0x00000001 | 0x00000001 | 0x00000000 | 0xffffffff | 0x00000000 | 0x00400024 |

* Nhận xét**:**

Ta thấy ở trường hợp này khi giá trị của j ứng với thanh ghi $s2 = 1 (0x00000001) và giá trị của i ứng với thanh ghi $s1 = 2 (0x00000002) thì giá trị của thanh ghi $t0 sẽ thay đổi khi ta thực hiện xong lệnh slt. Vì lệnh slt được dùng để so sanh j<i, ở trường hợp này điều kiện trên đã được thỏa mãn nên giá trị của thanh ghi $t0 được chuyển thành 1. Và khi chương trình chạy đến lệnh bne sẽ thực hiện rẽ nhánh vì lúc này giá trị của thanh ghi $t0 không bằng 0 nữa. Chương trình sẽ thực hiện các lệnh sau: gán giá trị cho thanh ghi $t2 với giá trị là -1 (0xffffffff) và giá trị của thanh ghi $t3 = $t3 + $t3. Lúc này nếu để ý thanh ghi pc, ta sẽ thấy từ lệnh slt sang lệnh bne, giá trị của thanh ghi tăng 16, có điều này là do khi thực hiện rẽ nhánh ta đã nhảy qua 4 lệnh xen kẽ giữa bne và và bắt đầu của else, mặc dù không thực hiện 4 lệnh đó nhưng máy tính vẫn đếm đủ 4 lệnh đó.

**\*Kết luận:**

Ta nhận thấy rằng chương trình chạy đúng so với kịch bản. Khẳng định rằng quá trình thực hiện chương trình của CPU là đúng so với lý thuyết đã học.

**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).

\***Trường hợp 1: i=0, n=6, step=1 (i<n và các bước nhảy sẽ không làm cho i vượt quá n)**

A screenshot of a computer

Description automatically generated

* Nhận xét :

Khi chạy đoạn chương trình trên, địa chỉ của các thanh ghi $s1-$s5, $t1, $t2 có sự thay đổi. Cụ thể như sau: đoạn chương trình trên là chương trình thực hiện tính tổng của các phần tử trong mảng. Trong đó: $s1 được sử dụng để lưu giá trị của chỉ số i, $s2 được dùng để lưu con trỏ trỏ đến phần tử đầu tiên của mảng, $s3 được dùng để lưu giá trị của n, $s4 được dùng để lưu giá trị của step, $s5 được dùng để lưu giá trị tổng, $t1 được dùng để lưu địa chỉ của phần tử thứ i trong mảng arr, $t0 được sử dụng để lưu giá trị của phần tử thứ i trong mảng. Chương trình chạy như sau:

+ Câu lệnh ‘slt $t2, $s1, $s3’ so sánh xem nếu $s1 < $s3 hay i < n thì giá trị $t2 = 1 nếu không thì bằng 0

+Câu lệnh ‘beq $t2, $zero, endloop’, Nếu như $t2 không bằng 0 thì sẽ tiếp tục nhãn loop, nếu không, thì thực hiện rẽ nhánh endloop.

+Câu lệnh ‘add $t1, $s1, $s1’ và ‘add $t1, $t1, $t1’ được sử dụng để tính 4\*i, vì các giá trị của mảng arr có độ dài 4 bytes.

+Câu lệnh ‘add $t1, $t1, $s2’ câu lệnh này được sử dụng để lấy địa chỉ của phần tử thứ i ở trong mảng và lưu vào t1. Phải sử dụng câu lệnh này vì ta sẽ truy cập vào các phần tử trong mảng dựa trên địa chỉ của các phần tử.

+Câu lệnh ‘lw $t0,0($t1)’ lưu giá trị của phần tử thứ i trong mảng vào thanh ghi $t0.

+Câu lệnh ‘add $s5, $s5, $t0’ cộng thêm giá trị của phần tử thứ i trong mảng vào sum.

+Câu lệnh ‘add $s1, $s1, $s4’ tăng giá trị của biến index lên bằng đúng với giá trị của step.

+Câu lệnh ‘j loop’ để nhảy về nhãn loop.

Chương trình chạy đúng với kịch bản, và tổng của mảng 1+3+5+6+2+4 = 21 và được lưu vào thanh ghi $s5.

Đối với thanh ghi pc ta nhận thấy khi kết thúc chương trình thanh ghi tăng một lượng 0x34(52) = 4x13 (với 4 là lương giá trị mà thanh ghi pc tăng khi thực hiện một lệnh, 13 là tổng số lệnh của chương trình) so với giá trị ban đầu. Có điều này là do khi thực hiện chương trình: lệnh la được quy đổi thành các lệnh lui và ori tương ứng.

A screenshot of a computer

Description automatically generated

**\*Trường hợp 2: i=7, n=6, step=1(i>n)**

**A screenshot of a computer program

Description automatically generated**

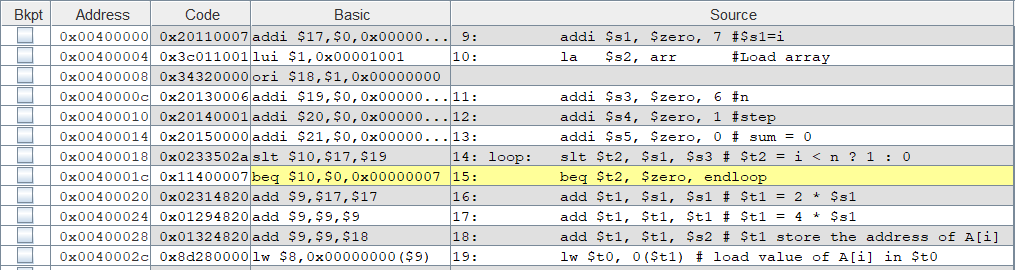
* Nhận xét:

Đoạn code trả về giá trị $s5 = 0 sau khi thực thi .

* Giá trị các thanh ghi

|  |  |
| --- | --- |
| Trước khi thực hiện lệnh | Sau khi thực hiện lệnh |
|  |  |

Ta nhận thấy giá trị của thanh ghi pc thay đổi một lượng 0x0000003c (60) tương đương với chương trình 4x15 tức là chương trình vẫn chạy 15 lệnh và dừng ở nhãn endloop như ảnh bên dưới. Xảy ra hiện tượng này là do giá trị ban đầu của i là 7 > n là 6, vì vậy khi thực hiện lệnh beq, máy tính thấy giá trị của $t2 = 0 nên đã nhảy đến endloop .



**\*Trường hợp 3 i=0, n=6, step=10(i<n nhưng bước nhảy sẽ khiến cho i > n)**

**A screenshot of a computer code

Description automatically generated**

* Nhận xét:

Đoạn code trả về giá trị $s5 = 1 sau khi thực thi .

* Bảng giá trị của các thanh ghi:

|  |  |
| --- | --- |
| Trước khi thực thi lệnh | Sau khi thực thi lệnh |
|  |  |

Ta nhận thấy giá trị của thanh ghi pc thay đổi một lượng 0x0000003c (60) tương đương với chương trình 4x15 tức là chương trình vẫn chạy 15 lệnh và dừng ở nhãn endloop như ảnh bên dưới. Xảy ra hiện tượng này là do giá trị ban đầu của i là 0 < n là 6, vì vậy máy tính sẽ tính giá trị đầu tiên của mảng là 1 và lưu vào $s5 , sau đó giá trị của i là 10 > n là 6, vì vậy khi thực hiện lệnh beq, thấy giá trị của $t2 = 0 nên đã nhảy đến endloop .

A screenshot of a computer

Description automatically generated

**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 program

Description automatically generated

* Nhận xét:

Đoạn code trên miêu tả cú pháp của switch case trong MIPS cụ thể như sau: đầu tiên ta gán giá trị của test bằng 1 số nguyên, sau đó so sánh test với lần lượt các case và thực hiện dựa trên kết quả của phép so sánh.

\*Trường hợp 1 test = 1 (thỏa mãn case\_1)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trạng thái chạy | | Nội dung của thanh ghi | | | | | | | |
| $s0 | $s1 | $s2 | $t0 | $t1 | $t2 | pc | $at |
| Ban đầu | | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000000 | 0x00000000 |
| Chạy lệnh la | Chạy lệnh lui | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000004 | 0x10010000 |
| Chạy lệnh ori | 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000008 | 0x10010000 |
| Chạy lệnh lw | | 0x10010000 | 0x00000001 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x4000000c | 0x10010000 |
| Chạy lệnh li (1) | | 0x10010000 | 0x00000001 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000010 | 0x10010000 |
| Chạy lệnh li (2) | | 0x10010000 | 0x00000001 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000000 | 0x40000014 | 0x10010000 |
| Chạy lệnh li (3) | | 0x10010000 | 0x00000001 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000018 | 0x10010000 |
| Chạy beq (1) | | 0x10010000 | 0x00000001 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x4000001c | 0x10010000 |
| Chạy beq (2) | | 0x10010000 | 0x00000001 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000030 | 0x10010000 |
| Chạy sub của Case 1 | | 0x10010000 | 0x00000001 | 0xffffffff | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000034 | 0x10010000 |
| Chạy j continue | | 0x10010000 | 0x00000001 | 0xffffffff | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000040 | 0x10010000 |

Lúc này lệnh beq1 sẽ trả về 0 và lệnh beq2 sẽ trả về 1 và thực hiện rẽ nhánh về case 1. Sau khi kết thúc chương trình, giá trị của thanh ghi $s2 là 0xffffffff.

\*Trường hợp 2 test = 0 (thỏa mãn case\_0)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trạng thái chạy | | Nội dung của thanh ghi | | | | | | | |
| $s0 | $s1 | $s2 | $t0 | $t1 | $t2 | pc | $at |
| Ban đầu | | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000000 | 0x00000000 |
| Chạy lệnh la | Chạy lệnh lui | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000004 | 0x10010000 |
| Chạy lệnh ori | 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000008 | 0x10010000 |
| Chạy lệnh lw | | 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x4000000c | 0x10010000 |
| Chạy lệnh li (1) | | 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000010 | 0x10010000 |
| Chạy lệnh li (2) | | 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000000 | 0x40000014 | 0x10010000 |
| Chạy lệnh li (3) | | 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000018 | 0x10010000 |
| Chạy beq (1) | | 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000028 | 0x10010000 |
| Chạy addi của case 0 | | 0x10010000 | 0x00000000 | 0x00000001 | 0x00000000 | 0x00000001 | 0x00000002 | 0x4000002c | 0x10010000 |
| Chạy j continue | | 0x10010000 | 0x00000000 | 0x00000001 | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000040 | 0x10010000 |

Lúc này lệnh beq1 sẽ trả về giá trị 1 và chương trình thực hiện rẽ nhánh đến case 1 và sau khi kết thúc chương trình, giá trị của thanh ghi $s2 sẽ bằng 0x00000001.

\*Trường hợp 3 test = 2 (thỏa mãn case\_2)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trạng thái chạy | | Nội dung của thanh ghi | | | | | | | |
| $s0 | $s1 | $s3 | $t0 | $t1 | $t2 | pc | $at |
| Ban đầu | | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000000 | 0x00000000 |
| Chạy lệnh la | Chạy lệnh lui | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000004 | 0x10010000 |
| Chạy lệnh ori | 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000008 | 0x10010000 |
| Chạy lệnh lw | | 0x10010000 | 0x00000002 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x4000000c | 0x10010000 |
| Chạy lệnh li (1) | | 0x10010000 | 0x00000002 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000010 | 0x10010000 |
| Chạy lệnh li (2) | | 0x10010000 | 0x00000002 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000000 | 0x40000014 | 0x10010000 |
| Chạy lệnh li (3) | | 0x10010000 | 0x00000002 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000018 | 0x10010000 |
| Chạy beq (1) | | 0x10010000 | 0x00000002 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x4000001c | 0x10010000 |
| Chạy beq (2) | | 0x10010000 | 0x00000002 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000020 | 0x10010000 |
| Chạy beq (3) | | 0x10010000 | 0x00000002 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000038 | 0x10010000 |
| Chạy add của case 2 | | 0x10010000 | 0x00000002 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x4000003c | 0x10010000 |
| Chạy j continue | | 0x10010000 | 0x00000002 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000040 | 0x10010000 |

Lúc này giá trị của beq1 và beq2 sẽ trả về 0, giá trị của beq3 sẽ trả về 1 và chương trình chuyển đến case\_2 tương ứng. Sau khi kết thúc chương trình, giá trị của thanh ghi $s2 bằng 0x00000000.

\*Trường hợp 4 test = 3 (tức là không có case nào thỏa mãn)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trạng thái chạy | | Nội dung của thanh ghi | | | | | | | |
| $s0 | $s1 | $s2 | $t0 | $t1 | $t2 | pc | $at |
| Ban đầu | | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000000 | 0x00000000 |
| Chạy lệnh la | Chạy lệnh lui | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000004 | 0x10010000 |
| Chạy lệnh ori | 0x10010000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000008 | 0x10010000 |
| Chạy lệnh lw | | 0x10010000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x4000000c | 0x10010000 |
| Chạy lệnh li (1) | | 0x10010000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000010 | 0x10010000 |
| Chạy lệnh li (2) | | 0x10010000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000000 | 0x40000014 | 0x10010000 |
| Chạy lệnh li (3) | | 0x10010000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000018 | 0x10010000 |
| Chạy beq (1) | | 0x10010000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x4000001c | 0x10010000 |
| Chạy beq (2) | | 0x10010000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000020 | 0x10010000 |
| Chạy beq (3) | | 0x10010000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000024 | 0x10010000 |
| Chạy j default | | 0x10010000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x00000001 | 0x00000002 | 0x40000040 | 0x10010000 |

Lúc này cả beq1, beq2, beq3 đều trả về 0 và chương trình sẽ chạy phần lệnh default.

**Assignment 4.**

Modify the Assignment 1, so that the condition tested is

a.i = j

b. i >= j

c. i+j <= 0

d. i+j > m+n

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 code  Description automatically generated |

1. i+j > m+n

|  |
| --- |
| A screenshot of a computer  Description automatically generated |

**Assignment 5.**

Modify the Assignment 2, so that the condition tested at the end of the loop is

a. i < n

b. i <= n

c. sum >= 0

d. A[i] == 0

1. i < n

|  |
| --- |
| A screenshot of a computer program  Description automatically generated |

1. i <= n

|  |
| --- |
| A screenshot of a computer program  Description automatically generated |

1. sum >= 0

|  |
| --- |
|  |

1. A[i] == 0

|  |  |
| --- | --- |
| |  | | --- | |  | |

**Assignment 6.**

Using all of above instructions and statements, create a new project to implement this function: find the element with the largest absolute value in a list of integers. Assuming that this list is store in an integer array and we know the number of elements in it.

A screenshot of a computer

Description automatically generated