**Laboratory Exercise 4**

**Arithmetic and Logical operation**

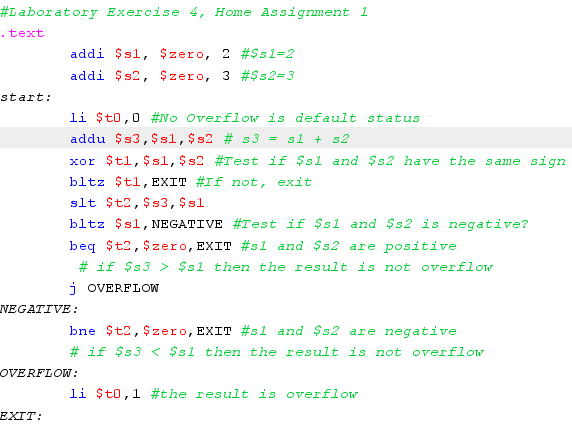
Họ và tên: Đỗ Mạnh Phương

MSSV: 20225660

**Assignment 1**

Create a new project to implement the Home Assigment 1. Compile and upload to simulator. Initialize two operands (register $s1 and $s2), run this program step by step, observe memory and registers value.

\*Trường hợp 1: $s1=2 và $s2= 3

****

**-**Bảng giá trị của các thanh ghi:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trạng thái chạy | Giá trị của các thanh ghi | | | | | | | |
| $s1 | $s2 | $s3 | $t0 | $t1 | $t2 | $t3 | Pc |
| Ban đầu | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000000 |
| Khởi tạo $s1 | 0x00000002 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000004 |
| Khởi tạo $s2 | 0x00000002 | 0x00000003 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000008 |
| Chạy lệnh li | 0x00000002 | 0x00000003 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x4000000c |
| Chạy lệnh addu | 0x00000002 | 0x00000003 | 0x00000005 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000010 |
| Chạy lệnh xor | 0x00000002 | 0x00000003 | 0x00000005 | 0x00000000 | 0x00000001 | 0x00000000 | 0x00000000 | 0x40000014 |
| Chạy lệnh bltz thứ 1 | 0x00000002 | 0x00000003 | 0x00000005 | 0x00000000 | 0x00000001 | 0x00000000 | 0x00000000 | 0x40000018 |
| Chạy lệnh slt | 0x00000002 | 0x00000003 | 0x00000005 | 0x00000000 | 0x00000001 | 0x00000000 | 0x00000000 | 0x4000001c |
| Chạy lệnh bltz thứ 2 | 0x00000002 | 0x00000003 | 0x00000005 | 0x00000000 | 0x00000001 | 0x00000000 | 0x00000000 | 0x40000020 |
| Chạy lệnh beq | 0x00000002 | 0x00000003 | 0x00000005 | 0x00000000 | 0x00000001 | 0x00000000 | 0x00000000 | 0x40000030 |

* Nhận xét : Khi chạy lệnh xor, giá trị thanh ghi $t1 đã trả về giá trị 1 (tức giá trị của thanh ghi này lớn hơn 0 (tức là cùng dấu). Vì thế, khi chạy lệnh bltz thứ 1, trình biên dịch chạy tiếp các lệnh tiếp theo mà không thoát ra. Khi chạy lệnh slt, do giá trị của thanh ghi $s3 lớn hơn thanh ghi $1, nên giá trị thanh ghi $t2 đã trả về giá trị 0. Khi chạy tiếp lệnh bltz thứ 2, do giá trị thanh ghi $s1 dương, nên trả về giá trị 0, không bị rẽ nhánh. Cuối cùng, do giá trị thanh ghi $t2 = 0, nên beq trả về giá trị true, và thoát chương trình. Như vậy, ta có được một phép tính 2 số dương và không bị hiện tượng overflow.

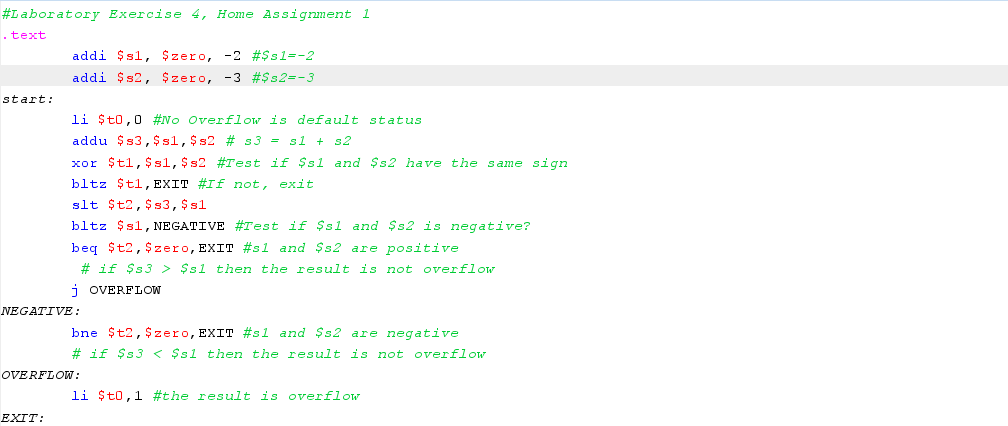
\*Trường hợp 2: Đặt giá trị thanh ghi $s1, $s2 là 2147483647 và 1

-Bảng giá trị:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trạng thái chạy | Nội dung của thanh ghi | | | | | | | |
| $s1 | $s2 | $s3 | $t0 | $t1 | $t2 | $t3 | Pc |
| Ban đầu | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000000 |
| Khởi tạo $s1 | 0x7fffffff | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x4000000c |
| Khởi tạo $s2 | 0x7fffffff | 0x00000001 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000010 |
| Chạy lệnh li | 0x7fffffff | 0x00000001 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000014 |
| Chạy lệnh addu | 0x7fffffff | 0x00000001 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000018 |
| Chạy lệnh xor | 0x7fffffff | 0x00000001 | 0x00000000 | 0x00000000 | 0x7ffffffe | 0x00000000 | 0x00000000 | 0x4000001c |
| Chạy lệnh bltz thứ 1 | 0x7fffffff | 0x00000001 | 0x00000000 | 0x00000000 | 0x7ffffffe | 0x00000000 | 0x00000000 | 0x40000020 |
| Chạy lệnh slt | 0x7fffffff | 0x00000001 | 0x00000000 | 0x00000000 | 0x7ffffffe | 0x00000001 | 0x00000000 | 0x40000024 |
| Chạy lệnh bltz thứ 2 | 0x7fffffff | 0x00000001 | 0x00000000 | 0x00000000 | 0x7ffffffe | 0x00000001 | 0x00000000 | 0x40000028 |
| Chạy lệnh beq | 0x7fffffff | 0x00000001 | 0x00000000 | 0x00000000 | 0x7ffffffe | 0x00000001 | 0x00000000 | 0x4000002c |
| Chạy j OVERFLOW | 0x7fffffff | 0x00000001 | 0x00000000 | 0x00000000 | 0x7ffffffe | 0x00000001 | 0x00000000 | 0x40000034 |
| Chạy lệnh li thứ 2 | 0x7fffffff | 0x00000001 | 0x00000000 | 0x00000001 | 0x7ffffffe | 0x00000001 | 0x00000000 | 0x40000038 |

* Tương tự như trường hợp 1, nhưng khi chạy đến lệnh slt, do giá trị thanh ghi $s3 nhỏ hơn giá trị thanh ghi $s1 nên giá trị thanh ghi $t2 đã trả về giá trị 1. Khi chạy lệnh beq, do giá trị thanh ghi $t2 khác 0, nên đã không thoát chương trình mà chạy tiếp. Như vậy, ta đã thấy hiện tượng overflow.

\*Trường hợp 3: Đặt giá trị thanh ghi $s1, $s2 là -2 và -3



-Bảng giá trị:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trạng thái chạy | Nội dung của thanh ghi | | | | | | | |
| $s1 | $s2 | $s3 | $t0 | $t1 | $t2 | $t3 | pc |
| Ban đầu | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000000 |
| Khởi tạo giá trị $s1 | 0xfffffffe | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000004 |
| Khởi tạo giá trị $s2 | 0xfffffffe | 0xfffffffd | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000008 |
| Chạy lệnh li | 0xfffffffe | 0xfffffffd | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x4000000c |
| Chạy lệnh addu | 0xfffffffe | 0xfffffffd | 0xfffffffb | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000010 |
| Chạy lệnh xor | 0xfffffffe | 0xfffffffd | 0xfffffffb | 0x00000000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x40000014 |
| Chạy lệnh bltz thứ 1 | 0xfffffffe | 0xfffffffd | 0xfffffffb | 0x00000000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x40000018 |
| Chạy lệnh slt | 0xfffffffe | 0xfffffffd | 0xfffffffb | 0x00000000 | 0x00000003 | 0x00000001 | 0x00000000 | 0x4000001c |
| Chạy lệnh bltz thứ 2 | 0xfffffffe | 0xfffffffd | 0xfffffffb | 0x00000000 | 0x00000003 | 0x00000001 | 0x00000000 | 0x40000028 |
| Chạy lệnh bne | 0xfffffffe | 0xfffffffd | 0xfffffffb | 0x00000000 | 0x00000003 | 0x00000001 | 0x00000000 | 0x40000030 |

* Nhận xét : Tương tự trường hợp 1 nhưng là 2 số âm, khi chạy lệnh slt, do giá trị thanh ghi $s3 nhỏ hơn giá trị thanh ghi $s1, nên giá trị thanh ghi $t2 đã trả về 1. Khi chạy lệnh bltz thứ 2, do giá trị thanh ghi $s1 nhỏ hơn 0, nên đã rẽ nhánh sang NEGATIVE. Khi đó, ở lệnh bne, do giá trị thanh ghi $t2 khác 0 nên đã kết thúc chương trình.

\*Trường hợp 4: Đặt giá trị thanh ghi $s1, $s2 là -2147483647 và -3

A screenshot of a computer program

Description automatically generated

-Bảng giá trị:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trạng thái chạy | Nội dung của thanh ghi | | | | | | | |
| $s1 | $s2 | $s3 | $t0 | $t1 | $t2 | $t3 | pc |
| Ban đầu | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000000 |
| Khởi tạo giá trị $s1 | 0x80000001 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000004 |
| Khởi tạo giá trị $s2 | 0x80000001 | 0xfffffffd | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000008 |
| Chạy lệnh li | 0x80000001 | 0xfffffffd | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x4000000c |
| Chạy lệnh addu | 0x80000001 | 0xfffffffd | 0xfffffffb | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000010 |
| Chạy lệnh xor | 0x80000001 | 0xfffffffd | 0xfffffffb | 0x00000000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x40000014 |
| Chạy lệnh bltz thứ 1 | 0x80000001 | 0xfffffffd | 0xfffffffb | 0x00000000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x40000018 |
| Chạy lệnh slt | 0x80000001 | 0xfffffffd | 0xfffffffb | 0x00000000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x4000001c |
| Chạy lệnh bltz thứ 2 | 0x80000001 | 0xfffffffd | 0xfffffffb | 0x00000000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x40000028 |
| Chạy lệnh bne | 0x80000001 | 0xfffffffd | 0xfffffffb | 0x00000000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x40000030 |
| Chạy lệnh li thứ 2 | 0x80000001 | 0xfffffffd | 0xfffffffb | 0x00000000 | 0x00000003 | 0x00000000 | 0x00000000 | 0x40000038 |

* Nhận xét: Tương tự trường hợp 3, tuy vậy khi chạy slt, do giá trị thanh ghi $s3 lớn hơn thanh ghi $s1, nên giá trị thanh ghi $t2 trả về 0. Tiếp tục chạy đến lệnh bne, do giá trị thanh ghi $t2 bằng 0 nên đã không kết thúc chương trình. Khi đó trình biên dịch chạy lệnh li thứ 2, nạp giá trị cho thanh ghi $t1 là 1, tương ứng với việc phép cộng trên đã bị hiện tượng overflow.

\*Trường hợp 5: Đặt giá trị thanh ghi $s1, $s2 là -2 và 3

A screenshot of a computer program

Description automatically generated

-Bảng giá trị:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trạng thái chạy | Nội dung của thanh ghi | | | | | | | |
| $s1 | $s2 | $s3 | $t0 | $t1 | $t2 | $t3 | pc |
| Ban đầu | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000000 |
| Khởi tạo giá trị $s1 | 0xfffffffe | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000004 |
| Khởi tạo giá trị $s2 | 0xfffffffe | 0x00000003 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000008 |
| Chạy lệnh li | 0xfffffffe | 0x00000003 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x4000000c |
| Chạy lệnh addu | 0xfffffffe | 0x00000003 | 0x00000001 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x40000010 |
| Chạy lệnh xor | 0xfffffffe | 0x00000003 | 0x00000001 | 0x00000000 | 0xfffffffd | 0x00000000 | 0x00000000 | 0x40000014 |
| Chạy lệnh bltz thứ 1 | 0xfffffffe | 0x00000003 | 0x00000001 | 0x00000000 | 0xfffffffd | 0x00000000 | 0x00000000 | 0x40000030 |

* Nhận xét : Việc nạp giá trị cho các thanh ghi $s1, $s2, $t0 vẫn như cũ, nhưng khi ta thực hiện lệnh xor, thanh ghi $t1 đã trở về giá trị < 0, do đó khi thực hiện lệnh bltz thứ 1 đã trả về giá trị TRUE và thoát ngay chương trình, tức 2 giá trị thanh ghi $s1 và $s2 đã được xác định trái dấu nhau.
* Như vậy, chương trình trên đã chạy đúng.

**Assignment 2**

Write a program to do the following tasks:

▪ Extract MSB of $s0

▪ Clear LSB of $s0

▪ Set LSB of $s0 (bits 7 to 0 are set to 1)

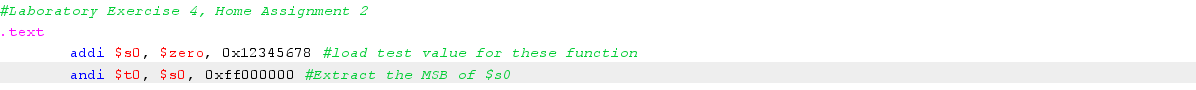
▪ Clear $s0 ($s0=0, must use logical instructions)

MSB: Most Significant Byte, LSB: Least Significant Byte

**A black text with numbers and arrows

Description automatically generated**

a)Extract MSB of $s0

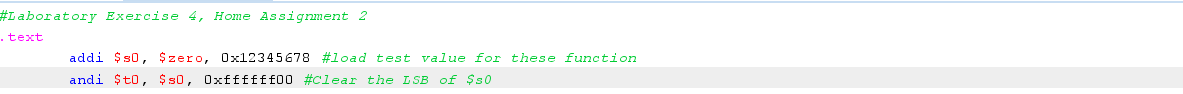


-Giá trị thanh ghi $t0:



-Nhận xét: giá trị thanh ghi $t0 là 0x12000000. Như vậy, chương trình đã chạy đúng.

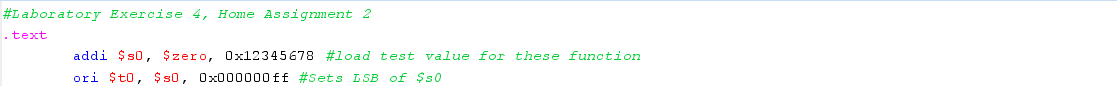
b) Clear LSB of $s0

 -Giá trị thanh ghi $t0:



-Nhận xét: giá trị thanh ghi $t0 là 0x12345600. Như vậy, chương trình trên đã chạy đúng.

c) Sets LSB of $s0

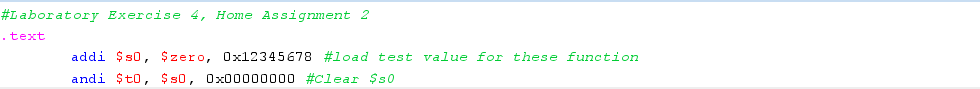


-Giá trị của thanh ghi $t0:



-Nhận xét: : thanh ghi $t0 có giá trị là 0x123456ff. Như vậy, chương trình trên đã chạy đúng.

d) Clear $s0

 -Giá trị của thanh ghi $t0:



-Nhận xét: thanh ghi $t0 có giá trị là 0x00000000. Như vậy, chương trình trên đã chạy đúng.

**Assignment 3**

Pseudo instructions in MIPS are not-directly-run-on-MIPS-processor instructions which need to be converted to real-instructions of MIPS. Re-write the following pseudo instructions using real instructions understood by MIPS processors:

a. abs $s0,$s1 $s0 <= | $s1 |

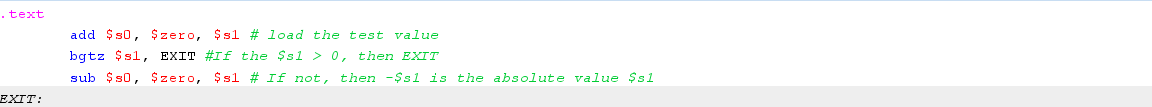
b. move $s0,$s1 $s0 <= $s1

c. not $s0, $s1 $s0 <= bit invert ($s1)

d. ble $s1,$s2,label

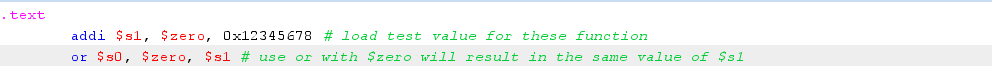
if ($s1 <= $s2) j label

1. Abs $s0, $s1



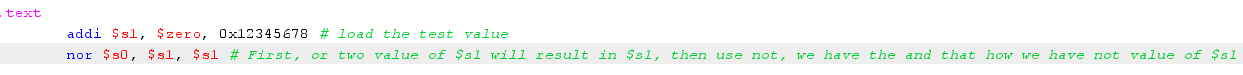
* Kết quả: Chương trình đã chạy đúng.

1. Move $s0, $s1



* Kết quả: Giá trị của thanh ghi $s0 là 0x12345678. Như vậy, chương trình đã chạy đúng.

1. Not $s0, $s1



* Kết quả: Giá trị của thanh ghi $s0 là 0xedcba987. Như vậy, chương trình trên đã chạy đúng.

1. Ble $s0, $s1

A screenshot of a computer

Description automatically generated

* Kết quả: chương trình đã chạy đúng.

**Assignment 4**

To dectect overflow in additional operation, we also use other rule than the one in Assignment 1. This rule is: when add two operands that have the same sign, overflow will occur if the sum doesn’t have the same sign with either operand. You need to use this rule to write another overflow detection program.

A screenshot of a computer

Description automatically generated

-TH1: Giá trị hai thanh ghi $s1, $s2 cùng dấu (giả sử là 2 và 3)

→ Giá trị của thanh ghi $s3 là 5, giá trị thanh ghi $t2 lớn hơn 0,, giá trị thanh ghi $t0 là 0, phép tính trên đã không bị tràn số.

-TH2: Giá trị hai thanh ghi $s1, $s2 trái dấu

+ Khi $s1 = -3, $s2 = 2 :

Khi đó, trình biên dịch sẽ chạy đến lệnh xor thứ nhất, cho ra giá trị thanh ghi $t2 lớn hơn 0. Từ đó, khi chạy lệnh bltz thứ 1 không bị rẽ nhánh, và thoát chương trình. Kết quả của thanh ghi $s3 là -1 và không có hiện tượng bị tràn số .

+ Khi $s1 = -2, $s3 =3 :

Chạy lệnh xor thứ 1, thanh ghi $t2 trả về giá trị 0. Do đó, khi chạy lệnh bltz thứ nhất sẽ rẽ nhánh sang ELSE. Khi đó ta tiếp tục kiểm tra dấu của $s2 và $s3. Vì chúng cùng dấu nên $t2 sẽ trả về giá trị lớn hơn 0, chương trình thoát ra, không bị hiện tượng tràn số.

- TH3: Giá trị hai thanh ghi cho hiện tượng tràn số 2147483647, 2

Khi đó, sau khi chạy bltz ở lệnh ELSE sẽ nhảy đến OVERFLOW do bị tràn số, khi đó $li sẽ có giá trị là 1, báo hiệu việc bị tràn số.

**Assignment 5**

Write a program that implement multiply by a small power of 2. (2, 4, 8, 16, etc for example)

A close-up of a document

Description automatically generated

* Nhận xét : Chương trình đã chạy đúng khi giá trị thanh ghi $s0 là 0x00000030 (48).