BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH – TUẦN 2

Họ và tên: Nguyễn Mạnh Tùng

MSSV: 20225682

Assignment 1: Lệnh gán số 16-bit

- Giá trị ban đầu của thanh ghi $s0: 0x00000000 và pc: 0x00400000
- Sau khi thực hiện câu lệnh: addi $s0, $zero, 0x3007

    $s0: 0x00003007

    pc: 0x00400004 (địa chỉ của câu lệnh tiếp theo)
- Sau khi thực hiện cây lệnh: add $s0, $zero, $0

    $s0: 0x00000000

    pc: 0x00400008
- So sánh mã máy với khuôn dạng lệnh

    + addi $s0, $zero, 0x3007: (I-type)

        opcode: 8 => 001000

        rt: $16 => 10000

        rs: $0 => 00000
        imm: 0x3007 => 0011 0000 0000 0111

    Mã máy: 0010 0000 0001 0000 0011 0000 0000 0111 => 0x20103007 (Trùng khớp)

    + add $s0, $zero, $0: (R-type)

        opcode: 000000

        rd: $16 => 10000

        rs: $0 => 00000

rt: $0 => 00000

shampt: 00000

funct: 32 => 100000

Mã máy: 0000 0000 0000 0000 1000 0000 0010 0000 => 0x00008020 (Trùng khớp)

- Sửa lại câu lệnh addi thành addi $s0, $zero, 0x2110003d:

+ Vì số 0x2110003d có khuôn dạng 32-bit => không thể gán trực tiếp => được tách thành 2 phần để thực hiện gán (2110 và 003d)

- 2110 được thực hiện gán bởi lệnh lui $1,0x0002110. Kết quả được lưu ở thanh ghi $at: 0x21100000
- 003d được thực hiện gán bở lệnh ori $1,$1,0x000003d. Kết quả được cập nhật ở thanh ghi $at: 0x2110003d
- Sau đó kết quả được gán về $0 từ $at => $s0: 0x2110003d
- Kết quả sau khi thực hiện dòng lệnh add $s0, $zero, $0 => $s0: 0x00000000

Assignment 2: lệnh gán số 32-bit

- Giá trị ban đầu của thanh ghi $s0: 0x00000000 và pc: 0x00400000

- Sau khi thực hiện câu lệnh lui $s0, 0x2110: cập nhật giá trị 4-bit bên trái của thanh ghi $s0. Giá trị của thanh ghi pc được cập nhật 0x00400004 (địa chỉ câu lệnh tiếp theo)
- Sau khi thực hiện câu lệnh ori $s0, 0x003d: cập nhật giá trị 4-bit bên phải của thanh ghi $s0. Gía trị của thanh ghi pc được cập nhật: 0x00400008

- Khi quan sát các byte trong vùng lệnh .text của cửa sổ Data Segment, nhận thấy các byte đầu tiên theo thứ tự trừ trái qua phải trùng khớp với các byte của cột Code ở cửa sổ Text Segment theo thứ tự từ trên xuống dưới.

Assigment 3: Lệnh gán (giả lệnh)

- Khi chạy chương trình, nhận thấy lệnh li được biên dịch thành các lệnh basic.

- Khi thực hiện câu lệnh li $s0, 0x2110003d (giá trị 32-bit) thì lệnh li được dịch thành cặp lệnh lui và ori được sử dụng để gán số 32-bit. => li là một giả lệnh có vai trò tương đương cặp lệnh basic lui và ori khi thực hiện gán số có giá trị >16-bit.
- Khi thực hiện câu lệnh li $s0, 0x2, lệnh li được dịch thành lệnh addiu, có chức năng gán giá trị nguyên, không tràn cho thanh ghi. => khi thực hiện lệnh gán số nguyên có giá trị =< 16-bit, giả lệnh li có chức năng tương đương lệnh addiu.

Assignment 4: Tính biểu thức 2x+y=?

- Kết quả chạy chương trình:

- Lệnh addi $t1, $zero, 5: gán giá trị cho thanh ghi $t1: 0x00000005
- Lệnh addi $t2, $zero, -1: gán giá trị cho thanh ghi $t2: 0xffffffff (giá trị được lưu của thanh ghi $t2 là kết quả sau khi đã lấy số bù hai của -1)
- Lệnh add $s0, $t1, $t1: thực hiện phép tính x+x=2x, giá trị của thanh ghi $s0: 0x0000000a
- Lệnh add $s0, $s0, $t2: thự hiện phép tính 2x+y, giá trị của thanh khi $s0: 0x00000009

=> kết quả chính xác

- Kiểm nghiệm khuôn mẫu lệnh I giữa hợp ngữ và mã máy:

- Câu lệnh addi $9, $0, 0x00000005:

  Mã máy: 0x20090005

  => 001000 00000 01001 0000000000000101

  001000 => 8 => addi: opcode

  00000 => 0 => $zero: rs

  01001 => 9 => $t1: rt

  0000000000000101 => 5 => imm

  => khuôn mẫu lệnh I: op, rt, rs, imm => addi $t1, $zero, 5 (Trùng khớp)

- Câu lệnh addi $10, $0, 0xffffffff:
  Mã máy: 0x200affff
  => 001000 00000 01010 1111111111111111
  001000 => 8 => addi: opcode
  00000 => 0 => $zero: rs
  01010 => 10 => $t2: rt
  1111111111111111 (số bù 2 của -1)=> -1 => imm

  => khuôn mẫu lệnh I: op, rt, rs, imm => addi $t2, $zero, -1 (Trùng khớp)

- Kiểm nghiệm khuôn mẫu lệnh R giữa hợp ngữ và mã máy:

- Câu lệnh add $16, $9, $9:
  Mã máy: 0x01298020
  => 000000 01001 01001 10000 00000 100000

  000000 => 0 => add: opcode

  01001 => 9 => $t1: rs

  01001 => 9 => $t1 :rt

  10000 => 16 => $s0: rd

  00000 => 0 => sh

  100000 => 32 => funct

  => khuôn mẫu lệnh R: op, rd, rs, rt => add $s0, $t1, $t1 (Trùng khớp)

- Câu lệnh add $16, $16, $10:
  Mã máy: 0x020a8020
  => 000000 10000 01010 10000 00000 100000

  000000 => 0 => add: opcode

  10000 => 16 => $s0: rs

  01010 => 10 => $t2 :rt

  10000 => 16 => $s0: rd

  00000 => 0 => sh

100000 => 32 => funct

=> khuôn mẫu lệnh R: op, rd, rs, rt => add $s0, $s0, $t2 (Trùng khớp)

Assignment 5: Phép nhân

- Sau khi thực hiện câu lệnh mul $s0, $s0, $3 xuất hiện thêm câu lệnh addi $1, $0, 0x00000003.

- Không thực hiện phép tính nhân trực tiếp với giá trị của thanh ghi $16 (vì không thể thực hiện nhân 3 số một lúc), nên thực hiện lưu giá trị 3 vào thanh ghi $at sau đó mới thực hiện phép nhân với giá trị của thanh ghi $16 và cho ra kết quả cuối cùng.

- Chạy chương trình:

- addi $t1, $zero, 4: $t1: 0x00000004
- addi $t2, $zero, 5: $t2: 0x00000005
- mul $s0, $t1, $t2: $s0: 0x00000014 và lo: 0x00000014
- mul $s0, $s0, 3: $at: 0x00000003
- mflo $s1: $s0, $s1 và lo có cùng value là 0x0000003c (có giá trị tập phân là 60 => kết quả đúng)

Assignment 6: Tạo biến và truy cập biến

- khi chạy chương trình, lệnh la được biên dịch thành cặp lệnh basic lui và ori

- Ở cửa sổ Labels, địa chỉa của ba biến X, Y, Z giống với hằng số khi biên dịch lệnh **la** thành mã máy

- Chạy chương trình

- Sau khi chạy lần 1:

+ Thanh $at từ 0x00000000 thay đổi thành 0x10010000

- Sau khi chạy lần 2:

+ Thanh $t8 thay đổi từ 0x00000000 thành 0x10010000

- Sau khi chạy lần 3:

+ Thanh $at giữ nguyên giá trị.

- Sau khi chạy lần 4:

+ Thanh $t0 thay đổi giá trị từ 0x00000000 thành 0x10010004

- Sau khi chạy lần 5:

+ Thanh $t1 thay đổi giá trị từ 0x00000000 thành 0x00000005

- Sau khi chạy lần 6:

+ Thanh $t2 thay đổi giá trị từ 0x00000000 thành 0xffffffff.

- Sau khi chạy lần 7:

+ Thanh $s0 thay đổi từ 0x00000000 thành 0x0000000a.

- Sau khi chạy lần 8:

+ Thanh $s0 thay đổi từ 0x0000000a thành 0x00000009

- Sau khi chạy lần 9, 10:

+ Thanh $t7 thay đổi từ 0x00000000 thành 0x10010008.

- Vai trò của lệnh **lw** và **sw:**

+ lw: Tải word từ bộ nhớ

+ sw: Lưu trữ word trong bộ nhớ