**Bài 2. Tập lệnh, các lệnh cơ bản, các chỉ thị biên dịch**

Nguyễn Thành Duy

MSSV: 20235696

Assignment 1: Lệnh gán số nguyên nhỏ 12-bit

Kết quả sau khi chạy chương trình:

A screenshot of a computer

AI-generated content may be incorrect.

Sau câu lệnh này:



Thanh ghi so thay đổi giá trị như mong muốn:



Thanh ghi pc: tăng từ 0x00400000 -> 0x00400004

****

Tiếp theo:



Thanh ghi so thay đổi giá trị như mong muốn:



Thanh ghi pc: 0x00400004 -> 0x00400008



0x51200413

0101 0001 0010 |0000 0|000 |0100 0|001 0011

op: 001 0011

funct3: 000

* Addi

rd: 01000 => x8 => so

rs1: 00000 => x0 => zero

imm: 010100010010 => 512

* addi so, zero, 0x512 (đúng như khuôn dạng lệnh)

0x00000433

0000 000|0 0000 |0000 0|000 |0100 0|011 0011

op: 011 0011

funct3: 000

funct7: 00000

* add

rd: 01000 => x8 => so

rs1: 00000 => x0 => zero

rs2: 00000 => x0

* add so, xo, zero (đúng như khuôn dạng lệnh)

Sửa lại lệnh addi

Chương trình sẽ báo lỗi:



Giải thích:

Do giá trị 0x20232024 vượt quá giới hạn giới hạn addi chỉ dùng để biểu diễn số 12-bit nên chương trình sẽ báo lỗi và không biên dịch được.

Assignment 2: Lệnh gán số 32-bit

Sau câu lệnh này:



* Thanh ghi so:



* Thanh ghi pc:



Tiếp theo:



* Thanh ghi so:



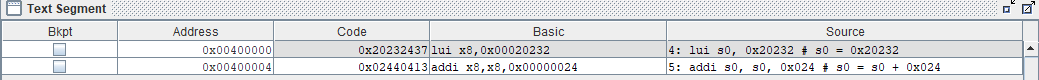
* Thanh ghi pc:



Ở bảng Data Segment:



Ở bảng Text Segment:



Ta thấy được rằng cách giá trị ở byte thứ nhất và thứ tư của địa chỉ 0x00400000 ở bảng Text Segment giống với mã máy(Code) ở bảng Text Segment.

Khi nạp một số 32-bit vào thanh ghi, nếu số 12-bit trong lệnh addi là số âm (bit thứ 11 bằng 1), thì cần mở rộng dấu thành 32 bit, và số 20-bit trong lệnh lui cần phải tăng lên 1. Giải thích tại sao?

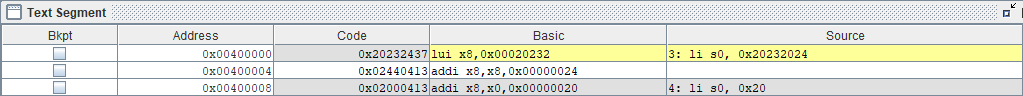
Giải thích:

Vì các Hằng số (immediate) trong RISC-V luôn là số bù 2 12-bit nên khi thực hiện cần mở rộng dấu thành 32-bits ở kiểu sign-extend.

Khi mở rộng dấu thành 32-bit để tránh tràn số thì ta bổ sung các F ở trước số cần mở rộng. Do vậy nên số 20-bit trong lệnh lui cần phải tăng lên 1 do các giá trị FF.. (-1) nên khi cộng ta chủ động cộng 1 vào số 20-bit.

Assignment 3: Lệnh gán (giả lệnh)

Sau khi chạy chương trình:



Quan sát các lệnh ở cột Source và cột Basic trong cửa sổ Text Segment ta thấy:

Câu lệnh: li so, 0x20232024 được tách thành 2 lệnh chính thống là:

lui x8, 0x00020232 và addi x8, x8, 0x00000024

Câu lệnh: li so, 0x20 được biên dịch câu lệnh:

addi x8, x0, 0x00000020

Thực chất: li là câu lệnh gán giá trị nhưng tùy vào giá trị cần gán thì sẽ có cách gán hay được biên dịch thành các câu lệnh chính thống khác nhau.

Assignment 4: Tính biểu thức 2x + y = ?

Kết quả sau khi chạy chương trình:

A screenshot of a computer

AI-generated content may be incorrect.

Câu lệnh đầu tiên: X = 0 + 5 = 5



Thanh ghi:



Câu lệnh thứ hai: Y = 0 – 1 = - 1



Thanh ghi:



Câu lệnh thứ ba: so = X + X = 5 + 5 = 10



Thanh ghi:



Câu lệnh thứ tư: so = 2X + Y = 10 – 1 = 9



Thanh ghi:



Kết thúc chương trình: Kết quả đúng như mong muốn

addi t1, zero, 5=> addi x6, x0, 5

op: 0010011

rd: x6 => 00110

funct3: 000

rs1: x0 => 00000

imm: 0000 0000 0101

* 0x00500313 (giống mã máy)

addi t2, zero, -1 => addi x7,x0, -1

op: 0010011

rd: x7 => 00111

funct3: 000

rs1: x0 => 00000

imm: 1111 1111 1111

* 0xfff00393 (giống mã máy)

Tương tự với: add

Assignment 5: Phép nhân

Câu lệnh thứ nhất:



* Thanh ghi:



Câu lệnh thứ hai:



* Thanh ghi:



Câu lệnh thứ ba:



* Thanh ghi:



Kết quả của đoạn chương trình đúng như mong muốn

Giải thích:

* Ở câu lệnh đầu tiên để gán giá trị 4 vào t1
* Ở câu lệnh thứ hai để gán giá trị 5 vào t2
* Ở câu lệnh thứ ba để tính tích t1 và t2 và lưu vào s1

Lệnh chia:

A screenshot of a computer

AI-generated content may be incorrect.

Tương tự như phép nhân thì chia dùng lệnh div

Kết quả của thanh ghi:







Kết quả s1 = 20 : 5 = 4 (giống với value ở thanh ghi)

Assignment 6: Tạo biến và truy cập biến

Kết quả sau khi chạy chương trình:

A screenshot of a computer

AI-generated content may be incorrect.

* Lệnh la được biên dịch thành 2 câu lệnh: auipc và addi

+ Địa chỉ tuyệt đối được chia thành hai phần: phần cao (20-bit) và phần thấp (12-bit). Kết hợp hai câu lệnh này để tải một địa chỉ đầy đủ 32-bit.

+ auipc: giúp lấy phần trên (20-bit) của địa chỉ nhãn dựa vào pc, auipc tính toán phần cao của địa chỉ label bằng cách lấy giá trị pc hiện tại và cộng thêm một giá trị offset.

+ addi: bổ sung phần thấp (12-bit) để hoàn chỉnh địa chỉ

* Giúp tải địa chỉ của nhãn (label) vào thanh ghi
* Cách này giúp chương trình có thể chạy độc lập với vị trí của bộ nhớ
* Dòng lệnh đầu tiên:



Khi chạy câu lệnh: Thanh ghi t5 có value ở thanh ghi là địa chỉ của X(đã định nghĩa ở trên)



* Dòng lệnh thứ hai:



Tương tự như t5 thì t6 có value ở thanh ghi là địa chỉ của Y.

* Dòng lệnh thứ ba:



Khi chạy câu lệnh: Giá trị value của t1 được lấy từ giá trị từ địa chỉ t5 mà giá trị của địa chỉ t5 là 5 (X = 5)



* Dòng lệnh thứ tư: tương tự với dòng lệnh thứ ba (lấy giá trị từ địa chỉ t6 gán cho value của t2)
* Dòng lệnh thứ năm:



* Dòng lệnh thứ sáu, bảy:



Dùng để cộng 2t1 và t2 và gán cho value của s0 (s0 = 2t1 + t2 = 10 – 1 = 9)



* Dòng lệnh thứ tám:



Được dùng để lấy địa chỉ của Z (xem trong cửa sổ Labels) và gán vào value của t4



* Dòng lệnh thứ chín:



Sau khi chạy câu lệnh thì sẽ lấy giá trị từ thanh ghi s0 và ghi vào bộ nhớ của địa chỉ t4:

Bộ nhớ trước khi chạy:

A screenshot of a computer

AI-generated content may be incorrect.

Bộ nhớ sau khi chạy:

A screenshot of a computer

AI-generated content may be incorrect.

Ở cửa số Labels:

A screenshot of a computer

AI-generated content may be incorrect.

* Vị trí của X, Y, Z trong bộ nhớ Data Segment:

X: 

Y: 

Z: 

Ta thấy giá trị của biến trong bộ nhớ giống với giá trị khởi tạo trong mã nguồn

* Vai trò câu lệnh lw, sw:

+) Về lw:

lw t1, o(t5) → Tải **giá trị tại địa chỉ t5** vào t1

(o(t5) là offset(độ dịch) tính từ t5)

lw t2, 0(t6) → Tải **giá trị tại địa chỉ t6** vào t2

(o(t6) là offset(độ dịch) tính từ t6)

Kết quả: t1 = X, t2 = Y

+) Về sw: Dùng để lưu một từ(32-bit) từ thanh ghi vào bộ nhớ

VD: # ao = 42 và s0 = 0x10010000

sw ao, o(so) # địa chỉ 0x10010000 sẽ chứa giá trị 42.

* Lệnh lb và sb:

+) Về lb: Nạp một byte (8-bit) từ bộ nhớ vào thanh ghi

+) Về sb: Lưu một byte (8-bit) từ thanh ghi vào bộ nhớ