Báo cáo tuần 2
Thực hành kiến trúc máy tính

Họ tên: Phan Minh Anh Tuấn
MSSV: 20205227

Mục lục

1 Home Assignment 1

1.1 Tên và ý nghĩa của 32 thanh ghi

Tên thanh ghi	Số hiệu thanh ghi	Công dụng
$zero	0	the constant value 0, chứa hằng số = 0
$at	1	assembler temporary, giá trị tạm thời cho hợp ngữ
$v0-$v1	2-3	procedure return values, các giá trị trả về của thủ tục
$a0-$a3	4-7	procedure arguments, các tham số vào của thủ tục
$t0-$t7	8-15	temporaries, chứa các giá trị tạm thời
$s0-$s7	16-23	saved variables, lưu các biến
$t8-$t9	24-25	more temporarie, chứa các giá trị tạm thời
$k0-$k1	26-27	OS temporaries, các giá trị tạm thời của OS
$gp	28	global pointer, con trỏ toàn cục
$sp	29	stack pointer, con trỏ ngăn xếp
$fp	30	frame pointer, con trỏ khung
$ra	31	procedure return address, địa chỉ trở về của thủ tục

Hình 1: Tên và ý nghĩa của 32 thanh ghi

1.2 Các thanh ghi đặc biệt PC, HI, LO

- pc : program counter, thanh ghi của CPU giữ địa chỉ của lệnh cần nhận vào để thực hiện
- hi : high-order word of multiply product, or divide reminder, lưu kết quả thao tác nhân của MIPS từ bit 32-63
- lo : low-order word of multiply product, or divide quotient, lưu kết quả thao tác nhân của MIPS từ bit 0-31

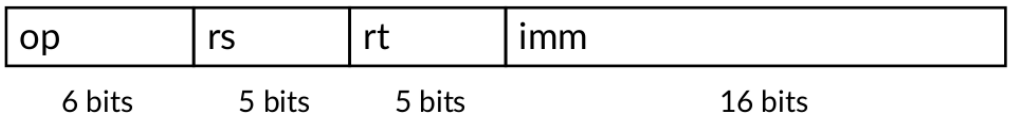
1.3 Khuôn dạng của 3 loại lệnh I, J, R

1.3.1 Lệnh kiểu R

Các trường của lệnh

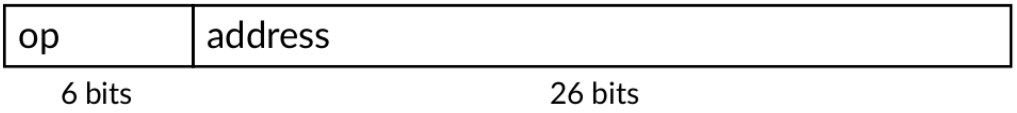
- op (operation code - opcode): mã thao tác. (với các lệnh kiểu R, op = 000000)
- rs: số hiệu thanh ghi nguồn thứ nhất
- rt: số hiệu thanh ghi nguồn thứ hai
- rd: số hiệu thanh ghi đích
- shamt (shift amount): số bit được dịch, chỉ dùng cho lệnh dịch bit, với các lệnh khác shamt = 00000
- funct (function code): mã hàm à mã hóa cho thao tác cụ thể

1.3.2 Lệnh kiểu I



Dùng cho các lệnh số học/logic với toán hạng tức thì và các lệnh load/store (nạp/lưu)

- rs: số hiệu thanh ghi nguồn (addi) hoặc thanh ghi cơ sở (lw, sw)
- rt: số hiệu thanh ghi đích (addi, lw) hoặc thanh ghi nguồn (sw)
- imm (immediate): hằng số nguyên 16-bit

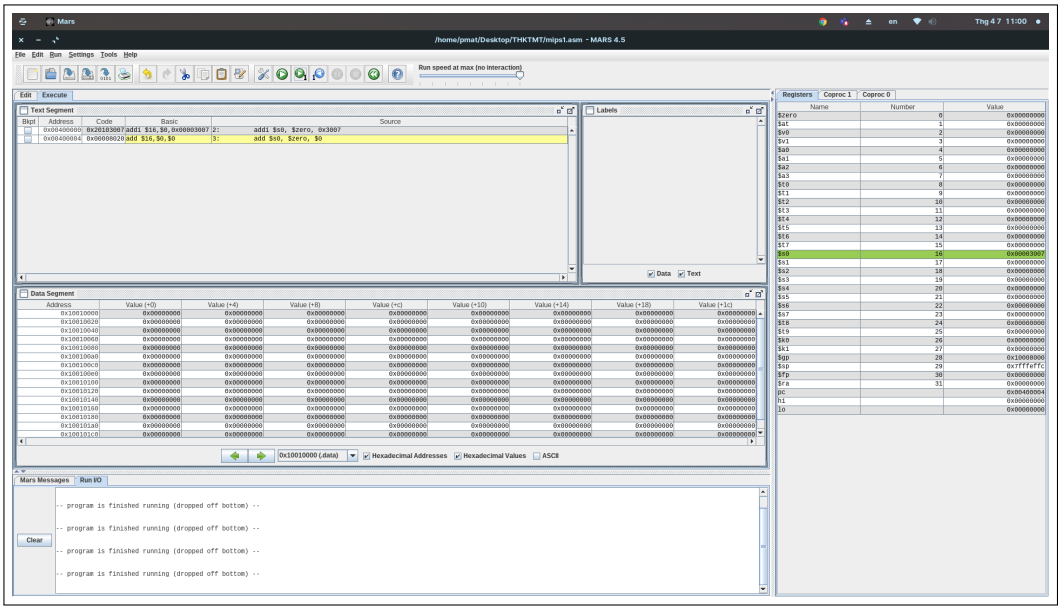


1.3.3 Lệnh kiểu J

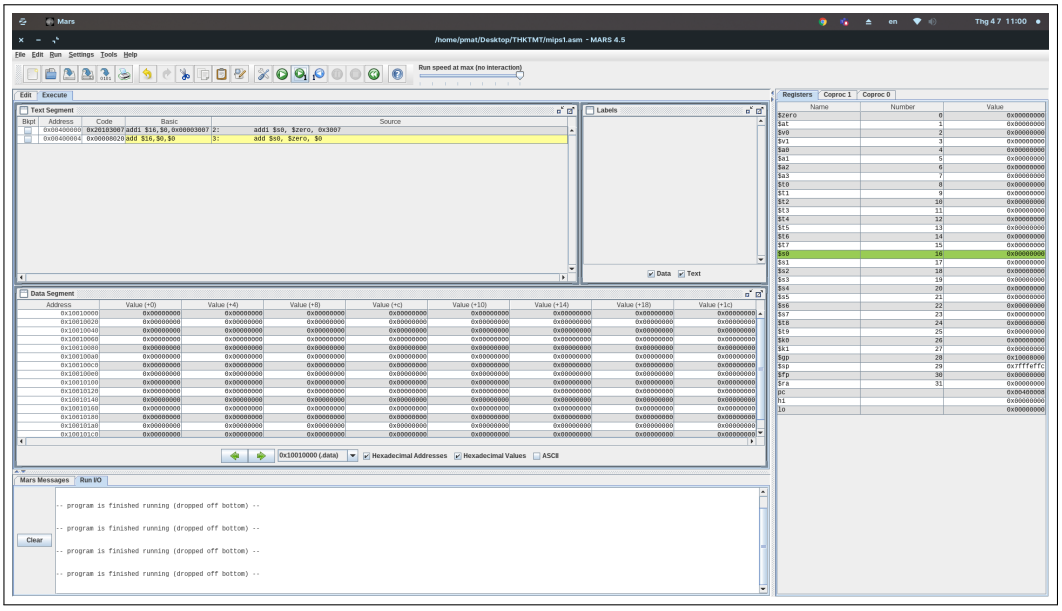
Toán hạng 26-bit địa chỉ Được sử dụng cho các lệnh nhảy

- j (jump) $\rightarrow$ op = 000010
- jal (jump and link) $\rightarrow$ op = 000011

2 Assignment 1: lệnh gán số 16-bit



Hình 2: addi $s0, $zero, 0x3007



Hình 3: add $s0, $zero, $0

Nhận xét:
Với thanh ghi $s0

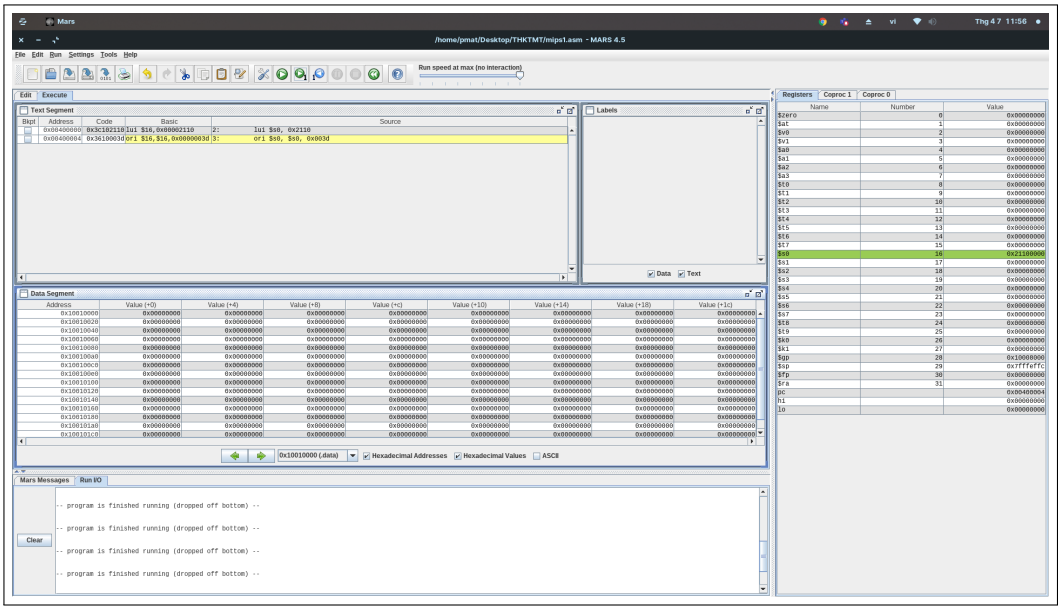
- Ở lần đầu tiên, thanh ghi $s0 thay đổi từ 0x00000000 thành 0x00003007 do toán tử addi lấy giá trị thanh ghi $zero cộng với giá trị tức thì 0x3007.
- Ở lần đầu thứ 2, thanh ghi $s0 thay đổi từ 0x00003007 về 0x00000000 do toán tử add nhận giá trị 2 thanh ghi $zero và $0, cộng giá trị số và lưu vào thanh ghi $s0

Với thanh ghi $pc: là thanh ghi dùng để giữ địa chỉ của lệnh được nhận vào

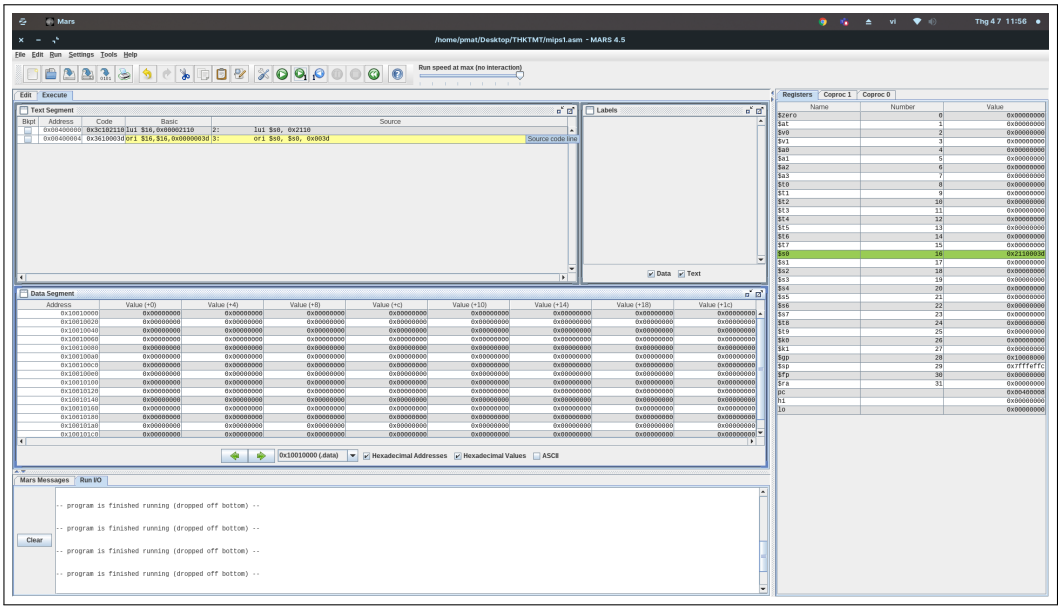
- Ở lần đầu tiên, lệnh được nhận vào có địa chỉ 0x00400000 nên địa chỉ thanh ghi $pc là 0x00400000.
- Ở lần thứ 2, lệnh được nhận vào có địa chỉ 0x00400004 nên địa chỉ thanh ghi $pc là 0x00400004.
- Ở lần thứ 3, tiếp tục tăng từ 0x00400004 lên 0x00400008, sau đó dừng lại.

Sửa lại lệnh lui thành: addi $s0, $zero 0x2110003d
Thanh ghi $s0 thành 0x2110003d do toán tử addi lấy giá trị thanh ghi $zero cộng với giá trị tức thì 0x2110003d. Do 0x2110003d là immediate 32 bits, nên tách thành toán tử lui, ori, add.

3 Assignment 2: lệnh gán số 32-bit



Hình 4: lui $s0, 0x2110



Hình 5: ori $s0, $s0, 0x003d

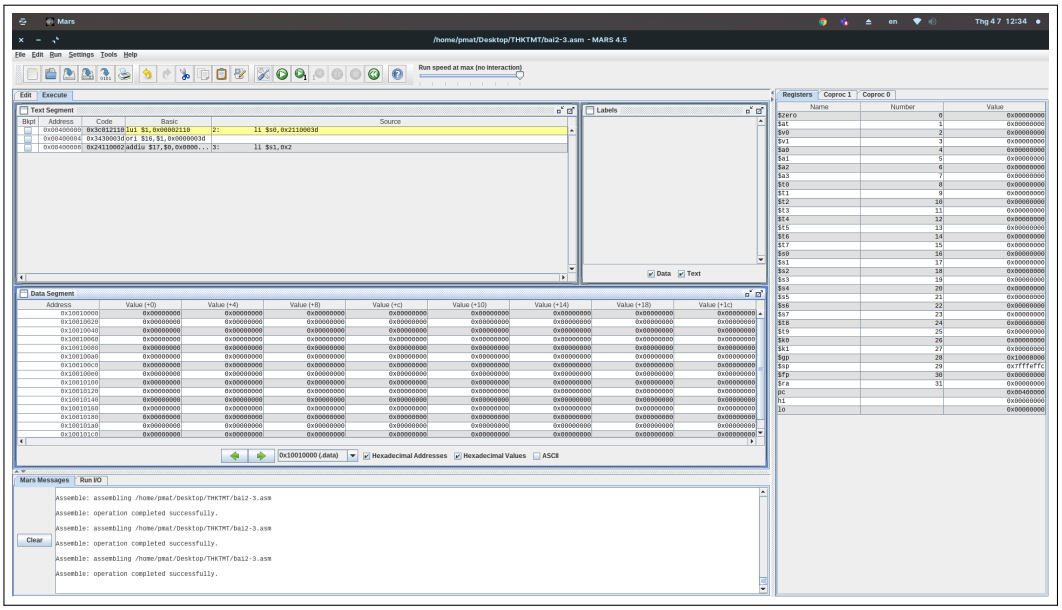
Nhận xét:
Với thanh ghi $s0

- Ở lần đầu tiên, thanh ghi $s0 thay đổi từ 0x00000000 thành 0x21100000 do toán tử lui nạp bit vào nửa trên.
- Ở lần đầu thứ 2, thanh ghi $s0 thay đổi từ 0x21100000 về 0x2110003d do toán tử ori cộng giá trị tức thời 0x003d

Với thanh ghi $pc: là thanh ghi dùng để giữ địa chỉ của lệnh được nhận vào

- Ở lần đầu tiên, lệnh được nhận vào có địa chỉ 0x00400000 nên địa chỉ thanh ghi $pc là 0x00400000.
- Ở lần thứ 2, lệnh được nhận vào có địa chỉ 0x00400004 nên địa chỉ thanh ghi $pc là 0x00400004.
- Ở lần thứ 3, tiếp tục tăng từ 0x00400004 lên 0x00400008, sau đó dừng lại.

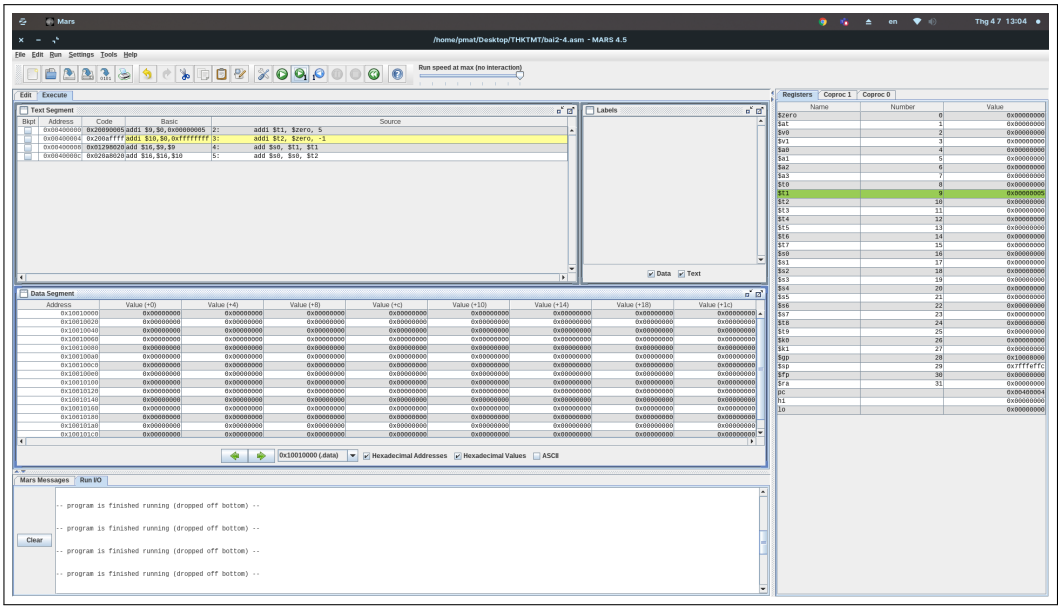
4 Assignment 3: lệnh gán (giả lệnh)



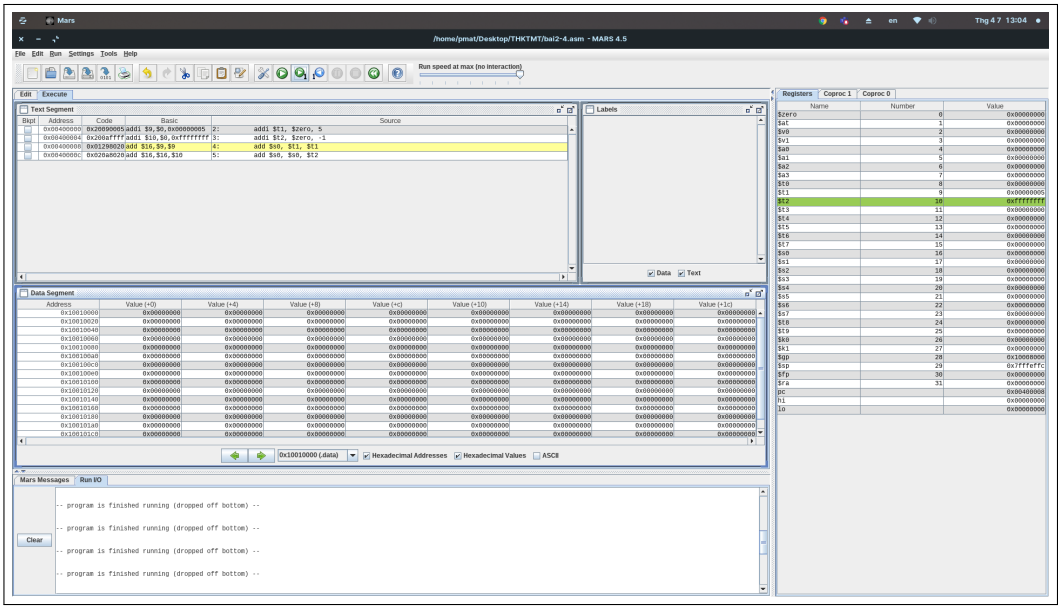
Hình 6: Lệnh gán Ass 3

Điều bất thường: Với số 32 bit như 0x2110003d khi gán vào $s0 sẽ được tách thành toán tử lui và ori, trong khi với số 4 bit như 0x2 thì sẽ dùng lệnh addiu mà không cần tách ra.
Giải thích: Do cơ chế nạp hằng số vào thanh ghi, với hằng số 32 bit sẽ tách ra lui và ori.

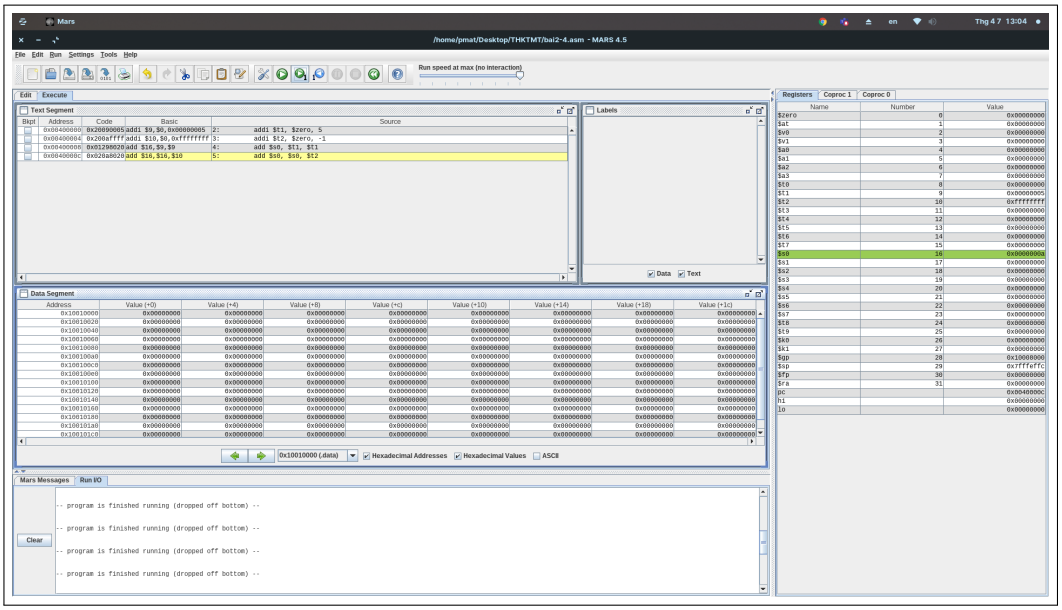
5 Assignment 4: tính biểu thức 2x + y = ?



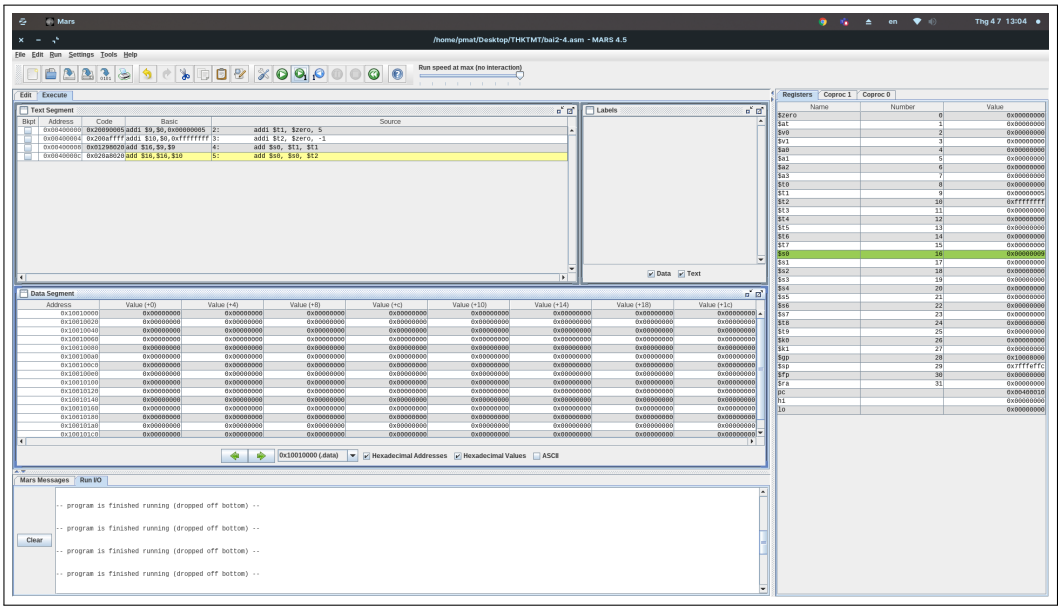
Hình 7: Câu lệnh gán thanh ghi $t1 = 5



Hình 8: Câu lệnh gán thanh ghi $t2 = 0-1

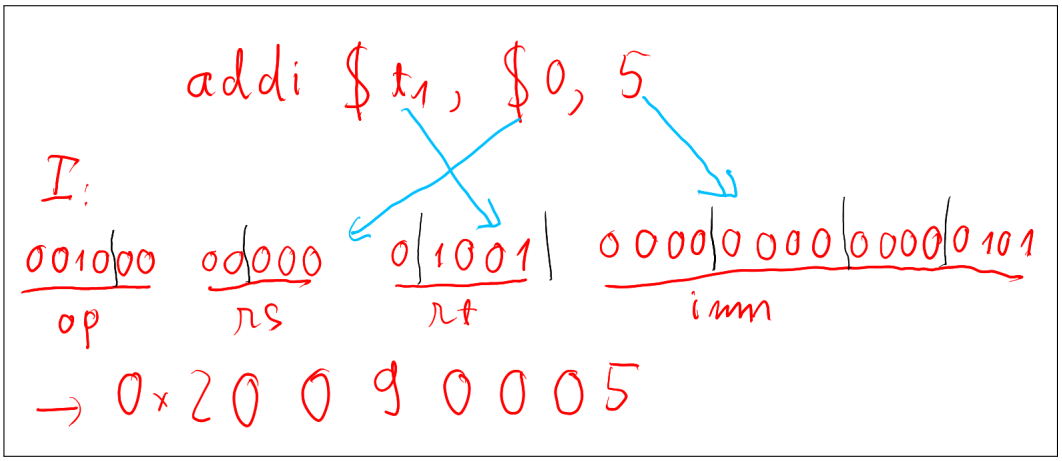


Hình 9: Câu lệnh gán thanh ghi $s0 = $t1 + $t1

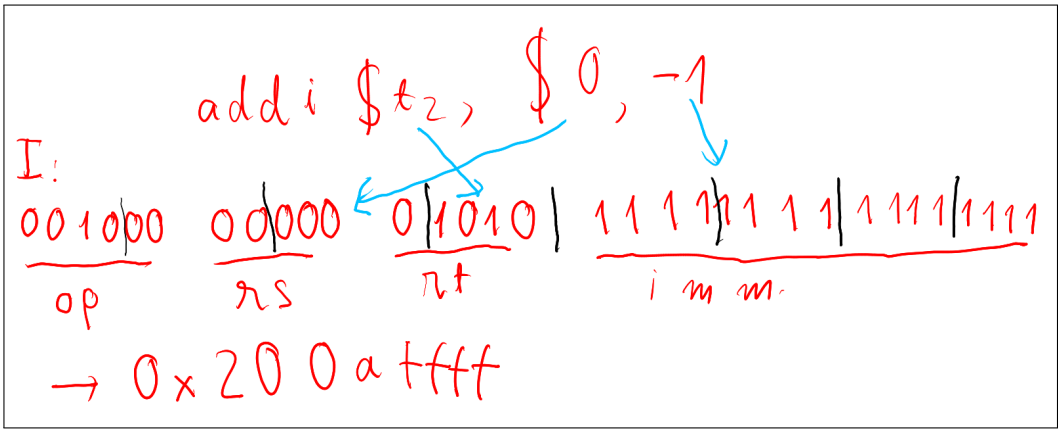


Hình 10: Câu lệnh gán thanh ghi $s0 = $s0 + $t2

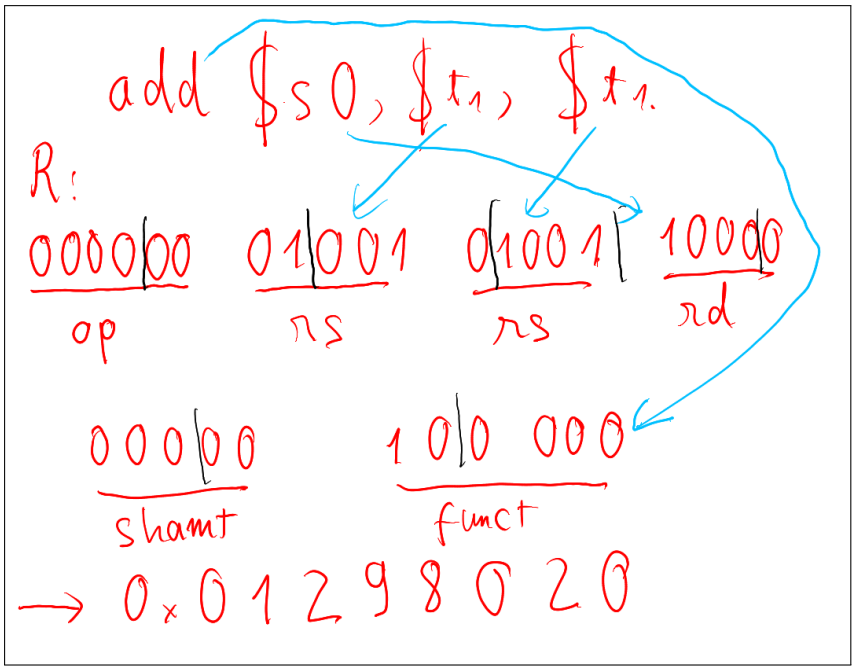
Kiểm tra điểm tương đồng với hợp ngữ và mã máy



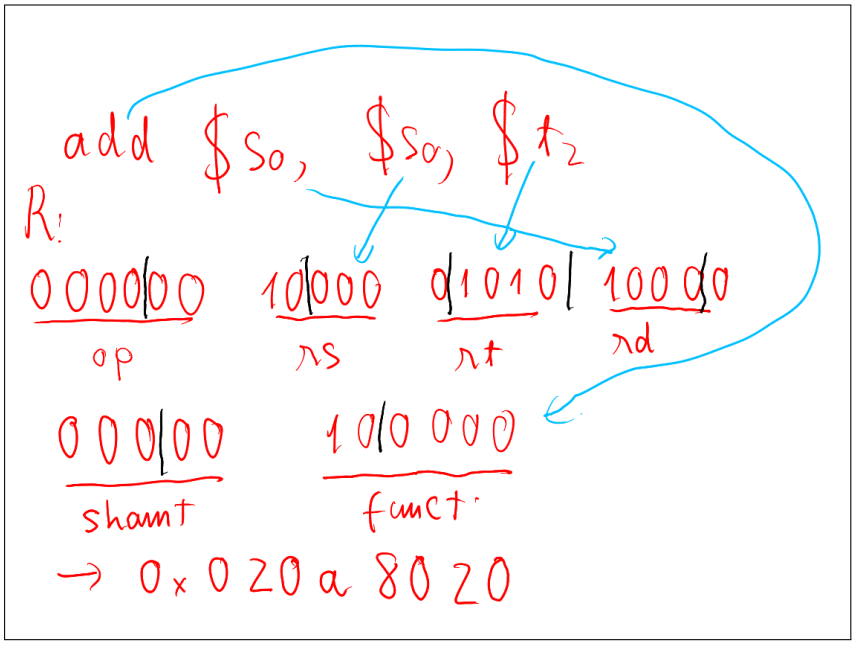
Hình 11: Câu lệnh addi $t1, $zero, 5, với khuôn mẫu kiểu lệnh I



Hình 12: Câu lệnh addi $t2, $zero, -1, với khuôn mẫu kiểu lệnh I

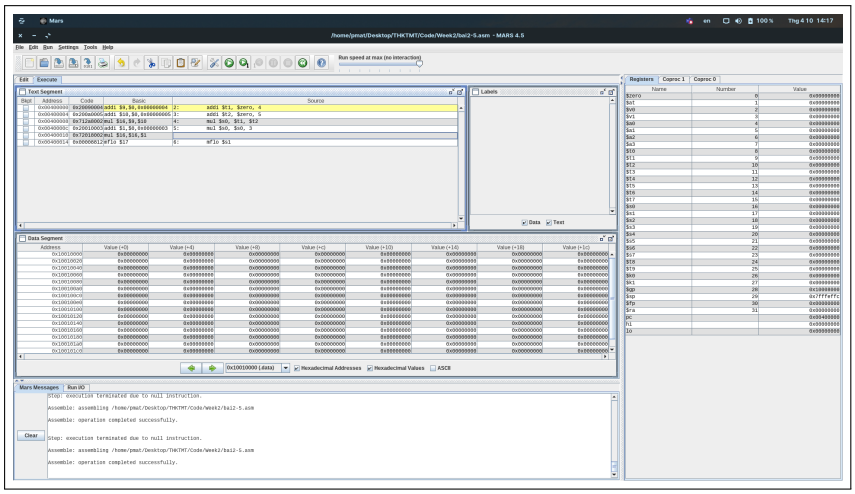


Hình 13: Câu lệnh add $s0, $t1, $t1, với khuôn mẫu kiểu lệnh R



Hình 14: Câu lệnh add $s0, $s0, $t2 với khuôn mẫu kiểu lệnh R

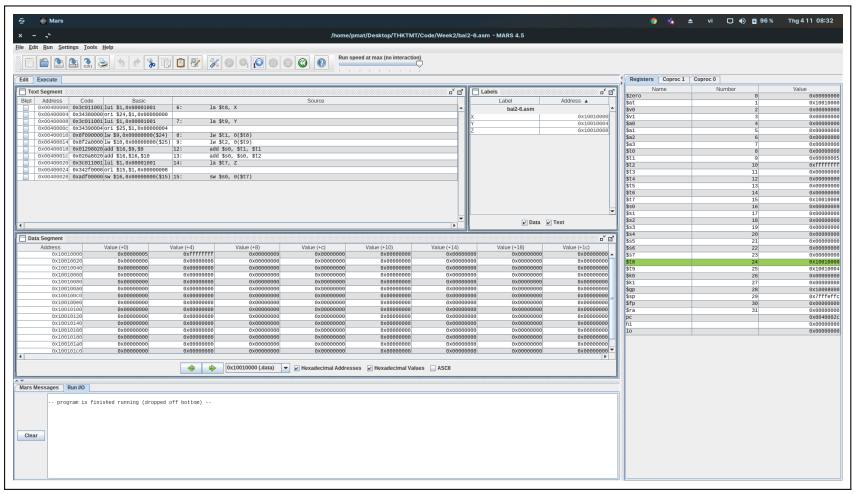
6 Assignment 5: phép nhân



Hình 15: Phép nhân

Điều bất thường: Tại bước mul ($s0, $s0, 3) không thực hiện phép nhân ngay mà chuyển 3 lên thanh ghi at rồi mới thực hiện nhân

7 Assignment 6: tạo biến và truy cập biến



Hình 16: Tạo biến và truy cập biến

Lệnh la được dịch thành lui và ori
lw: đọc word dữ liệu 32-bit từ bộ nhớ đưa vào thanh ghi (load word)
sw: ghi word dữ liệu 32-bit từ thanh ghi đưa ra bộ nhớ (store word)