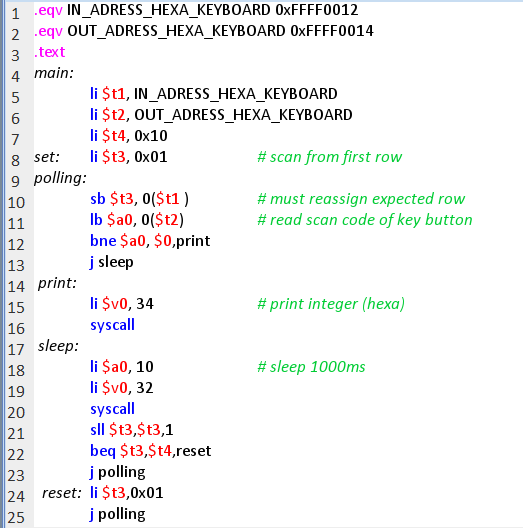
**Bài thực hành số 11**

**Lớp : 139365 – Học phần: Thực hành Kiến trúc máy tính**

**Họ và tên : Nguyễn Thị Thùy Dung MSSV : 20215009**

**Bài 1:**



*Thực hiện gõ chương trình vào công cụ* ***MARS***

* Kết quả :

**

* Giải thích :

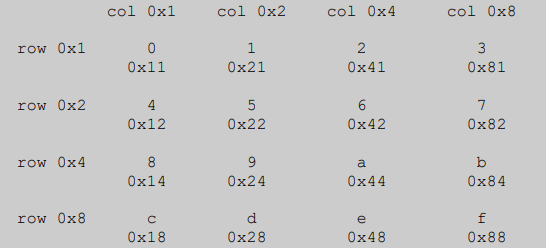
Gán giá trị của IN\_ADDRESS\_HEXA \_KEYBOARD là 0xFFFF0012, giá trị OUT\_ADDRESS\_HEXA\_KEYBOARD là 0xFFFF0014

Khởi tạo giá trị $t1 = 0XFFFF0012 , $t2 = 0XFFFF0014 , $t4 = 0x10

Thực hiện một vòng lặp polling với khởi tạo $t3 = 0x01, trước tiên store byte giá trị $t3 (lấy byte cuối củng $t3) vào ô nhớ có địa chỉ là giá trị $t1, tức là đang thực hiện duyệt hàng 1 bởi 4 bit cuối của $t3 là 0001. Tiếp theo load byte(lấy byte cuối), lấy giá trị từ ô nhớ có địa chỉ là giá trị $t2 vào $a0, khi đó lấy được phím đã được bấm, thực hiện so sánh nếu $a0 = $0 = 0(phím chưa được bấm) thì không thực hiện in, ngược lại in phím tương ứng ra màn hình

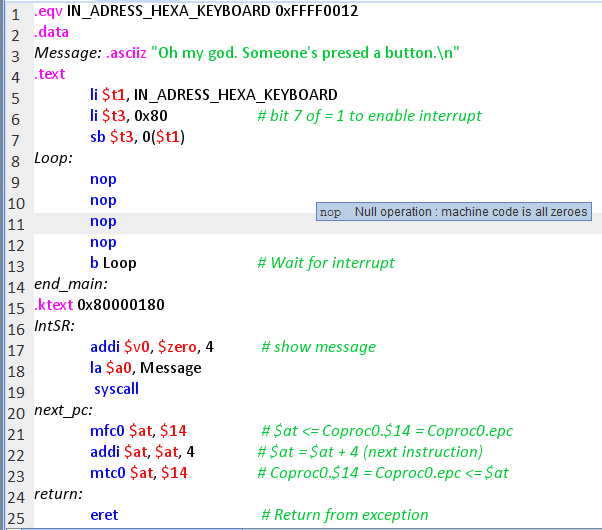
Hàm *print* thực hiện in phim được bấm ra màn hình với $v0 = 34

Hàm *sleep* gán $a0 = 10, $v0 = 32, syscall để nghỉ 10ms, sau đó thực hiện sll (dịch bit sang 1) $t3 để duyệt sang hàng 2,3,4 (0x02,0x04,0x08). Cứ mỗi lần dịch bit cần so sánh $t3 với $t4 = 0x10 hay chưa. Nếu bằng (tức là đã duyệt xong 4 hàng) thì thực hiện reset gán $t3 = 0x01 (quay lại hàng 1) và thực hiện *polling* , nếu không thì quay lại *polling* luôn để duyệt các hàng tiếp theo



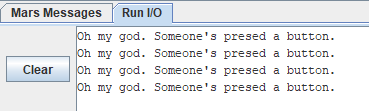
Khi ấn số 2 ở hàng 1, cột 3 trên màn hình sẽ hiện lên 0x00000041, hàng 1 suy ra 4 bit cuối là 0001, cột 3 nên 4 bit đầu của byte cuối là 0100

Bài 2 :



*Thực hiện gõ chương trình vào công cụ* ***MARS***

* Kết quả :



* Giải thích :

Khởi tạo IN\_ADDRESS\_HEXA\_KEYBOARD bằng 0xffff0012 đại diện địa chỉ cho bàn phím, khởi tạo Message : ‘Oh my god. Someone’s pressed a button’

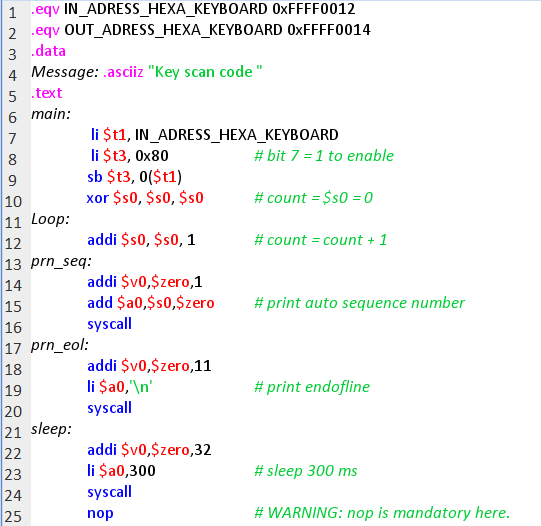
Gán giá trị $t1 = 0xffff0012 , $t3 = 0x80 với bit 7 = 1 để thực hiện ngắt. Sau đó thực hiên lưu giá trị $t3 vào địa chỉ có giá trị là $t1, bit 7 bằng 1 giúp kích hoạt ngắt từ bàn phím.

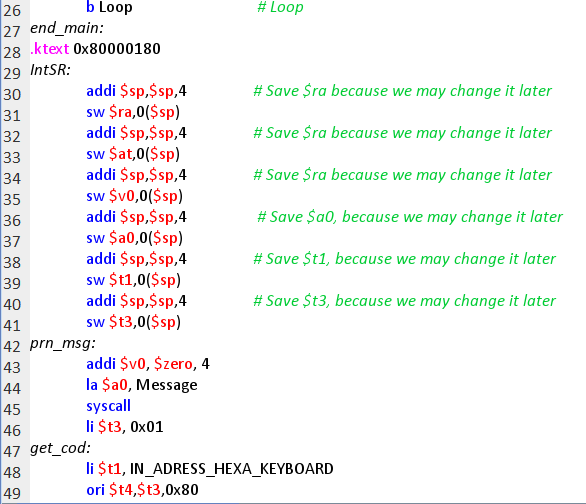
Hàm loop sẽ thực hiện vòng lặp của chương trình chính, chỉ khi xảy ra ngắt (người dùng ấn phím trên digital lab sim) thì sẽ nhảy đến chương trình con ngắt .ktext ở địa chỉ bắt đầu 0x80000180. Chương trình con ngắt gán $v0 = 4, $a0 chứa địa chỉ message để in ra màn hình ai đó đã ấn phím, syscall để message được in ra màn hình.

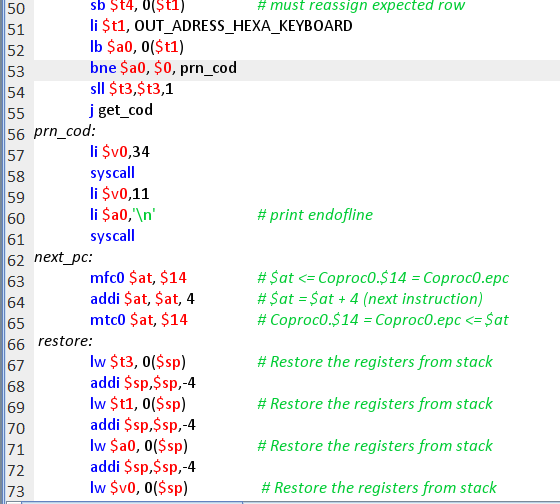
Do thanh ghi PC vẫn chứa địa chỉ của lệnhmà ngắt xảy ra, tức là lệnh đã thực hiện xong, chứ không chứa địa chỉ củalệnh kế tiếp. Bởi vây phải tự lập trình để tăng địa chỉ chứa trong thanh ghiepc bằng cách sử dụng 2 lệnh mfc0 (để đọc thanh ghi trong bộ đồng xử lýC0 ra $at) tại dòng 21, tăng $at thêm 4 để nó chỉ đến địa chỉ lệnh kế tiếp và mtc0 (để ghi giá trị $at vào thanh ghi trong bộ đồng xử lý C0)

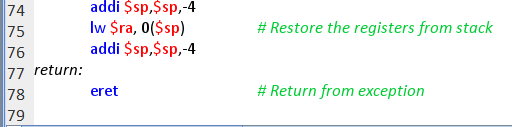
Câu lệnh return: eret đánh dấu sự kết thúc của chương trình ngắt và quay về chương trình chính để thực hiện lệnh tiếp theo

Bài 3 :



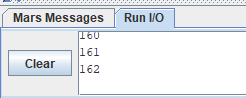






*Thực hiện gõ chương trình vào công cụ* ***MARS***

* Kết quả :



* Giải thích:

Khởi tạo IN\_ADDRESS\_HEXA\_KEYBOARD = 0xffff0012 đại diện cho địa chỉ bàn phím để đọc dữ liệu, OUT\_ADDRESS\_HEXA\_KEYBOARD = 0xffff0014 đại diện cho địa chỉ bàn phím để ghi dữ liệu, khai báo Message : ‘Key scan code’

Gán $t1 = 0xffff0012, $t3 = 0x80 (bit 7 =1 giúp kích hoạt chương trình ngắt), lưu giá trị byte ở $t3 vào địa chỉ có giá trị là $t1, khởi tạo $s0 = 0 (count = $s0 = 0), thực hiên vòng loop để chương trình chính. Vòng lặp này tăng $s0 = $s0 +1 (count++), gán $v0 =1 , $a0 = $s0 để in $s0 (count) ra màn hình bằng lệnh syscall, sau đó gán $v0 = 11, $a0 = ‘\n’ gọi syscall để xuống dòng, tiếp theo gán $v0 = 32, $a0 = 300 và syscall để thực hiện ngủ 300ms, sau đó tiếp tục thực hiện vòng lặp loop, chú ý sau syscall cần có lệnh *nop* nếukhông việc ghi nhận giá trị của thanh ghi PC vào EPC sẽ bị sai

Khi người dùng ấn bất kỳ 1 phím trên data lab sim thì chương trình ngắt sẽ được thực hiện, chương trình .ktext ở địa chỉ bắt đầu 0X80000180 được gọi đến. Trước tiên chương trình ngắt thực hiện lưu các giá trị tại các thanh ghi $ra, $at, $v0, $a0, $t1, $t3 trong stack bởi giá trị của các thanh ghi có thể bị thay đổi trong chương trình ngắt. Việc lưu trữ và stack được thực hiện bằng cách tăng lần lượt $sp đi 4 và thực hiện lưu giá trị các thanh ghi vào $sp

Chương trình ngắt gán $v0 = 4, $a0 là địa chỉ của Message thực hiên syscall để in ra thông báo 1 phím đã được bấm.

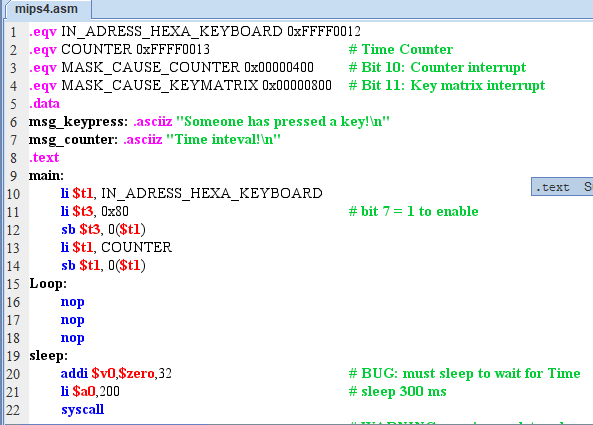
Đoạn lệnh get\_cod thực hiện duyệt toàn bộ data lab sim bằng cách khởi tạon $t3 = 0x01 (mã phím đầu tiên), thực hiên $t4 = $t3 or 0x80 (cần bit 7 bằng 1 để kích hoạt chương trình), lưu $t4 vào $t1 (input\_keyboard), sau đó gán $t1 = output\_keyboard và lấy giá trị từ địa chỉ $t1 gán vào $a0 để lấy địa chỉ phím đã đọc được, nếu $a0 bằng 0 (chưa phím nào được nhấn) thực hiện $t3 = $t3 \* 2 bằng lệnh sll để duyệt sang cột tiếp theo, và quay lại get\_cod . Cứ thực hiện duyệt tất cả các phím. Còn nếu $a0 khác 0 đã có phím được nhấn thì gán cho $v0 = 34 để in mã phím được ấn ra màn hình sau đó gán $v0 = 11, $a0 = ‘\n’ để xuống dòng

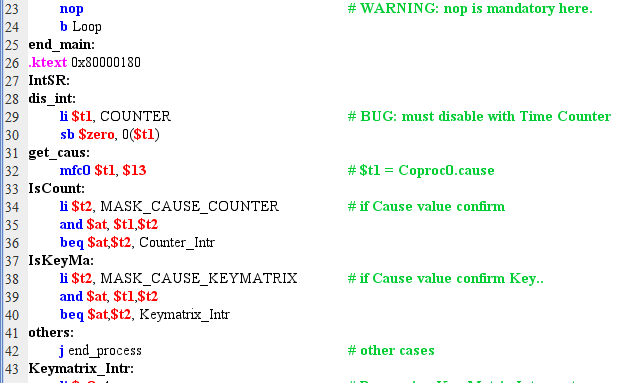
Do thanh ghi PC vẫn chứa địa chỉ của lệnhmà ngắt xảy ra, tức là lệnh đã thực hiện xong, chứ không chứa địa chỉ củalệnh kế tiếp. Bởi vây phải tự lập trình để tăng địa chỉ chứa trong thanh ghiepc bằng cách sử dụng 2 lệnh mfc0 (để đọc thanh ghi trong bộ đồng xử lýC0 ra $at) tại dòng 21, tăng $at thêm 4 để nó chỉ đến địa chỉ lệnh kế tiếp và mtc0 (để ghi giá trị $at vào thanh ghi trong bộ đồng xử lý C0)

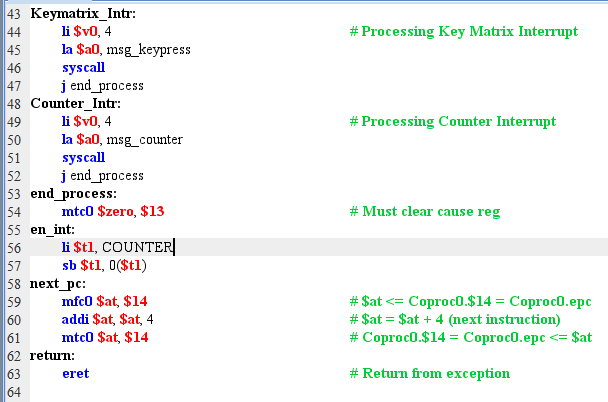
Sau đó thực hiện lấy lại các giá trị thanh ghi đã lưu trữ trước đó từ stack bằng cách lần lượt $sp = $sp - 4 và lấy giá trị từ $sp vào các thanh ghi

Cuối cùng là return eret đánh dấu sự kết thúc của chương trình ngắt và quay về chương trình chính để thực hiện lệnh tiếp theo

Bài 4:







*Thực hiện gõ chương trình vào công cụ* ***MARS***

* Kết quả :
* Giải thích :

Gán IN\_ADDRESS\_HEXA\_KEYBOARD bằng 0xffff0012, COUNTER bằng 0xffff0013, MASK\_CAUSE\_COUNTER bằng 0x00000400 (bởi vì bit 10 bằng 1 chính là ngắt bằng bộ đếm thời gian), MASK\_CAUSE\_KEYMATRIX bằng 0x00000800(bit 11 bằng 1 chính là ngắt bởi ma trận phím), khởi tạo msg\_keypress hiển thi thông báo khi có phím được ấn và msg\_counter hiển thị bộ đếm thời gian

Khai báo $t1 = 0xffff0012, $t3 = 0x80 (bit 7 = 1 để kích hoạt ma trận phím), sau đó lưu giá trị $t3 vào địa chỉ có giá trị là $t1, tiếp tục gán $t1 = 0xffff0013m sau đó tiếp tục lưu giá trị $t1 vào chính nơi có địa chỉ là giá trị của $t1. Chương trình hàm loop thực hiện các lệnh (nop), trong đó có gán $v0 = 32, $a0 = 200, syscall để thực hiện ngủ 200ms, tiếp theo là nop là là bắt buộc như đã giải thích ở assignment 3, sau đó branch lại loop.

Khi bấm một nút thì sẽ chạy đến chương trình con ngắt .ktext ở địa chỉ bắt đầu 0x80000180. Trước tiên gán $t1 = 0xffff0013 và gán 0 vào địa chỉ có giá trị là $t1, để tránh lỗi bộ đếm thời gian. Để xác định nguyên nhân ngắt do đâu, thực hiện lấy giá trị của $13 vào $t1.

Kiểm tra xem ngắt có do bộ đếm hay không ta khai báo $t2 = 0x00000400, and $t1 và $t2 nếu bằng nhau chính là ngắt do bộ đếm khi đó sẽ chạy nhãn IsCount in ra màn hình thông báo Time Interval , kiểm tra xem có ngắt do ma trận phím hay không cũng tương tự cho $t2 = 0x0000080, and $t1 và $t2, nếu $t1 = $t2 thì in ra màn hình thông báo Someone pressed key, nếu không thì ngắt xảy ra do chương thức khác và nhảy đến end\_process.

Ở đây gán 0 cho $13 để xóa phương thức ngắt cũ, thực hiện $t1 = 0xffff0013, lại lưu $t1 vào địa chỉ có giá trị là $t1, sau đó chỉnh thanh ghi $14 giống như đã giải thích ở assignment 2. Cuối cùng return eret để quay lại chương chính kết thúc chương trình ngắt