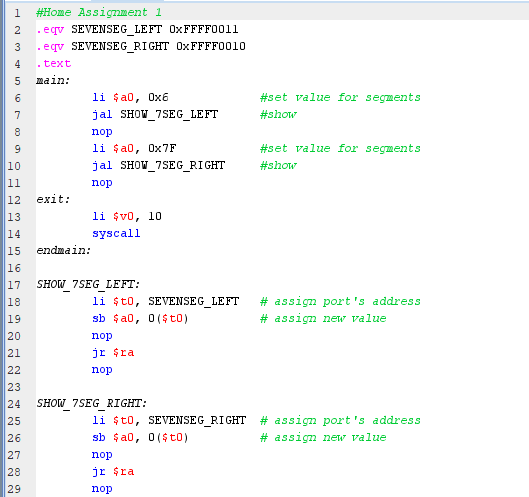
**Bài thực hành số 6**

**Lớp: 139365 – Học phần: Thực hành Kiến Trúc Máy Tính**

**Họ và tên: Đinh Thị Hồng Phúc MSSV: 20215118**

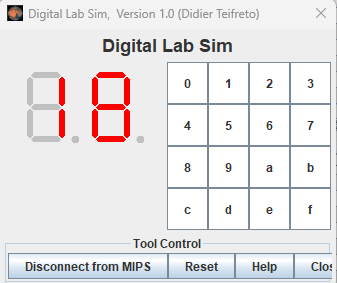
Bài 1.



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

Giải thích:

* Đầu tiên dùng *.eqv* để định nghĩa 2 hằng số, để lưu địa chỉ của 2 số trên LED 7 thanh (bên trái và bên phải)
* *main:* dùng để gán giá trị đầu vào vào thanh ghi $a0 đại diện cho giá trị mà các thanh bên trái, phải sẽ hiển thị qua 2 chương trình con *SHOW\_7SEG\_LEFT* và *SHOW\_7SEG\_RIGHT*
* *exit:* thoát chương trình
* *SHOW\_7SEG\_LEFT* tương tự *SHOW\_7SEG\_RIGHT:* Đầu tiên gán địa chỉ của màn hình LED 7 thanh tương ứng (trái, phải) vào thanh ghi $t0 (dòng 18 và 25). Sau đó gán giá trị trong $a0 vào địa chỉ của màn hình hiển thị LED 7 thanh (dòng 19 và 26). Lệnh *jr $ra* được dùng để trở về địa chỉ gọi hàm (trong *main*)



*Thực hiện chạy chương trình với* ***MARS***

Cách hiển thị số 18 trên LED 7 thanh:

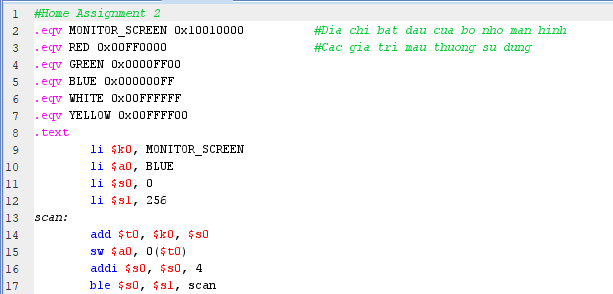
* Hiển thị 1: chỉ thanh b và c sáng → 0x6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| • | g | f | e | d | c | b | a |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

* Hiển thị 8: tất cả các thanh cùng sáng → 0x7F

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| • | g | f | e | d | c | b | a |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

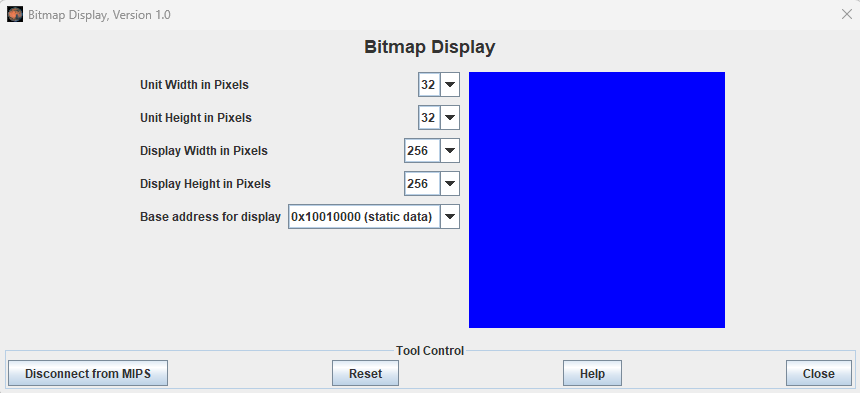
Bài 2.



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

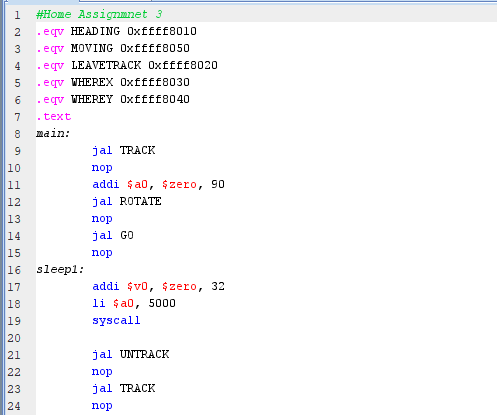
Giải thích:

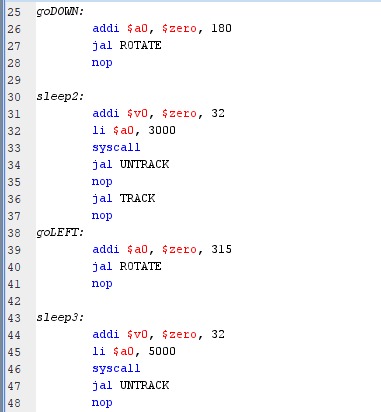
* Dòng 1: Xác định địa chỉ bắt đầu của bộ nhớ màn hình, nơi các pixel được lưu trữ
* Các dòng lệnh *.eqv* tiếp theo dùng để định nghĩa các hằng số cho các giá trị màu sắc
* Dòng 9, 10: Đặt giá trị của địa chỉ màn hình vào thanh ghi $k0, giá trị màu *blue* vào thanh ghi $a0
* Thanh ghi $s0 được gán giá trị bằng 0 dùng để lưu địa chỉ của từng điểm ảnh trên màn hình, $s1 gán bằng 256 là giới hạn số điểm ảnh
* Vòng lặp *scan* thực hiện duyệt qua từng điểm ảnh bằng cách tính địa chỉ của điểm ảnh hiện tại (cộng địa chỉ bắt đầu của màn hình - $k0 với giá trị $s0), lưu giá trị màu vào điểm ảnh tại địa chỉ hiện tại. Sau đó tăng giá trị $s0 lên 4 để trỏ đến điểm ảnh tiếp theo. Khi $s0 vẫn nhỏ hơn hoặc bằng $s1 thì tiếp tục vòng lặp

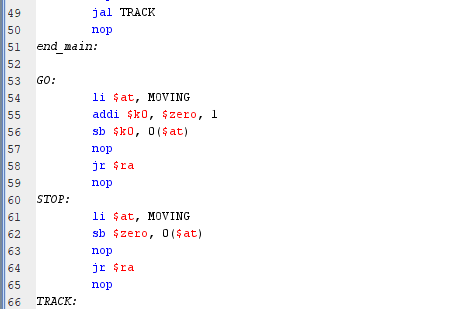


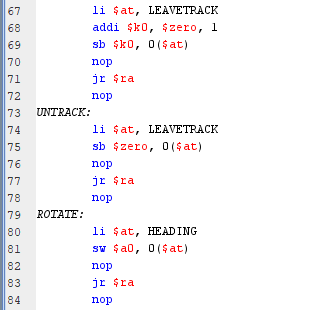
*Thực hiện chạy chương trình với* ***MARS***

Bài 3.





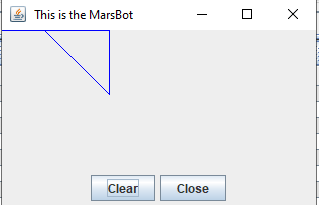




*Thực hiện gõ chương trình vào công cụ* ***MARS***

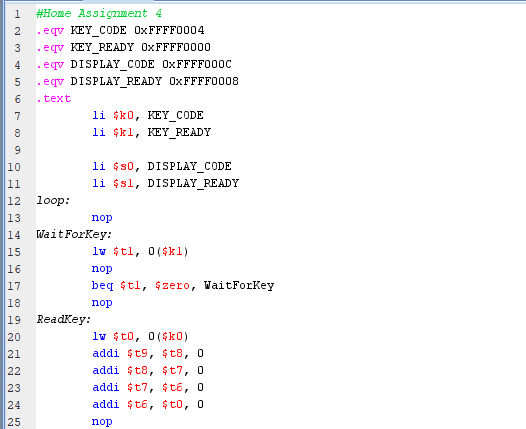
Giải thích:

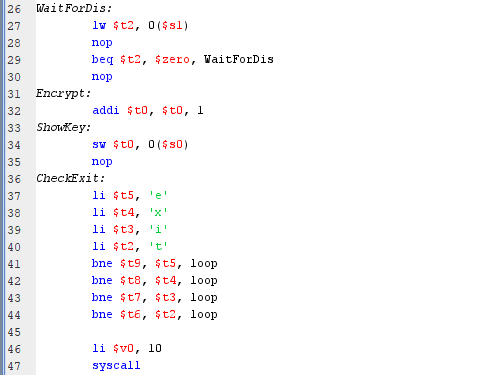
* *.eqv* được dùng để định nghĩa các hằng số *HEADING, MOVING, LEAVETRACK, WHEREX, WHEREY* đại diện cho các địa chỉ trong bộ nhớ
* *main:* gọi chương trình con *TRACK*. Chương trình con này bật cờ *LEAVETRACK* để ghi lại các bước của bot vẽ hình tam giác trên màn hình. Dòng 11 thiết lập giá trị góc xoay là 90 độ → vẽ 1 đường kẻ ngang. Dòng 12 gọi chương trình con *ROTATE,* chương trình con này lưu giá trị góc xoay vào địa chỉ *HEADING*. Tiếp theo gọi chương trình con *GO*, chương trình con này đặt cờ *MOVING* cho phép bot di chuyển
* *sleep1:* Dòng 17, giá trị 32 được gán vào thanh ghi $v0 để thực hiện chức năng delay. Dòng 18 gán trị trị 5000 vào thanh ghi $a0 đại diện cho số lượng đơn vị thời gian (microseconds) muốn bot vẽ đường thẳng. Chương trình con *UNTRACK* sẽ tắt cờ *LEAVETRACK, TRACK* sẽ bật cờ *LEAVETRACK* để ghi lại đường kẻ
* Các hướng khác của tam giác (kẻ xuống – 180 độ, kẻ cạnh huyền – 315 độ) tương tự như đường kẻ ngang với thời gian chờ (hàm *sleep*) tương ứng
* Các chương trình con như *GO, STOP, TRACK, UNTRACK, ROTATE*  sử dụng để tiếp tục di chuyển, dừng lại, chuyển sang dòng kẻ khác, xoay. Thao tác với các cờ và giá trị trong bộ nhớ để điều khiển bot



*Thực hiện chạy chương trình với* ***MARS***

Bài 4.

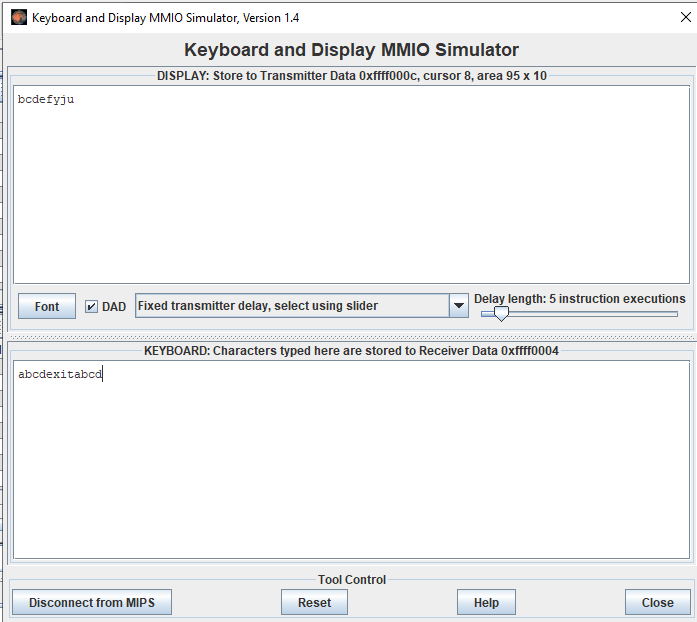




*Thực hiện gõ chương trình vào công cụ* ***MARS***

Giải thích:

* Các dòng lệnh *.eqv* định nghĩa các hằng số để lưu địa chỉ của các thanh ghi điều khiển bàn phím và màn hình
* Dòng 7-11: Đặt giá trị của các hằng số tương ứng vào thanh ghi $k0, $k1 (địa chỉ và trạng thái sẵn sàng của thanh ghi điều khiển bàn phím), $s0, $s1 (địa chỉ và trạng thái sẵn sàng của thanh ghi điều khiển màn hình)
* Vòng lặp *loop* đọc kí tự từ bàn phím, mã hoá và hiển thị trên màn hình
* Vòng lặp *WaitForKey* kiểm tra xem có kí tự nào được nhận từ bàn phím hay chưa. Nếu không, vòng lặp sẽ tiếp tục chờ (tải giá trị từ ô nhớ tại địa chỉ được lưu trong thanh ghi $k1 vào $t1, địa chỉ này đại diện cho trạng thái của bàn phím. Sau đó kiểm tra xem giá trị trong $t1 có bằng 0 không. Nếu bằng 0 thì bàn phím chưa sẵn sàng và vòng lặp tiếp tục chờ)
* Sau khi nhận được kí tự từ bàn phím, vòng lặp *ReadKey* được thực hiện để thực hiện các bước mã hoá và hiển thị. Đọc giá trị của thanh ghi điều khiển bàn phím vào $t0, giá trị kí tự được nhập từ bàn phím. Sao chép giá trị từ $t0 sang $t6, $t7, $t8, $t9
* Vòng lặp *WaitForDis* tương tự như *WaitForKey* dùng để kiểm tra xem màn hình đã sẵn sàng để hiển thị kí tự mới hay chưa. Nếu không sẽ tiếp tục chờ
* Khi màn hình sẵn sàng, vòng lặp *Encrypt* được thực hiện để thay đổi giá trị của kí tự $t0 trước khi hiển thị lên màn hình
* *ShowKey:* Lưu giá trị của $t0 (kí tự đã được mã hoá) vào thanh ghi điều khiển màn hình
* Vòng lặp *CheckExit*: Kiểm tra xem chuỗi nhập vào có phải “exit” hay không. Nếu không, vòng lặp sẽ tiếp tục ở đầu vòng *loop* (đặt giá trị ASCII của chuỗi “exit” vào từng thanh ghi sau đó so sánh giá trị các thanh ghi $t9 → $t6 với từng kí tự. Nếu có bất kì sự khác nhau nào sẽ nhảy đến *loop*)
* Cuối cùng gán $v0 bằng 10 để kết thúc chương trình



*Thực hiện chạy chương trình với* ***MARS***