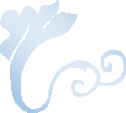
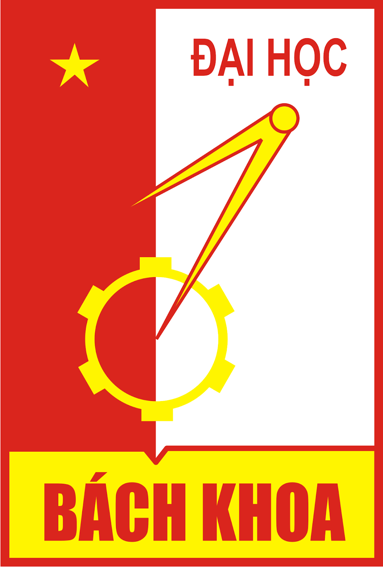


# HANOI UNIVERSITY OF SCIENCE & TECHNOLOGY SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY





FINAL PROJECT REPORT

**IT3280E – ASSEMBLY LANGUAGE AND COMPUTER ARCHITECTURE LAB**

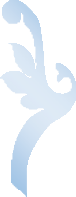
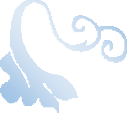


|  |  |  |
| --- | --- | --- |
|  | **Course information** |  |
| **Course ID** | **Course title** | **Class ID** |
| IT3280 | Assembly language and computer architecture lab | 147796 |

|  |  |  |
| --- | --- | --- |
| **Student information** | | |
| **Student’s full name** | **Class** | **Student ID** |
| Mai Minh Quân | Việt Nhật 03 | 20225661 |

Vũ Minh Quân Việt Nhật 04 20225910

**Instructor:** MSc. Đỗ Công Thuần



Bài 1: Vẽ hình trên màn hình Bitmap

1. Yêu cầu bài toán:

* Sử dụng MIPS để vẽ một quả bóng di chuyển trên màn hình mô phỏng Bitmap của Mars).
* Nếu quả bóng đập vào cạnh của màn hình thì sẽ di chuyển theo chiều ngược lại.
* Chiều di chuyển phụ thuộc vào phím người dùng bấm, gồm có (di chuyển lên (W), di chuyển xuống (S), Sang trái (A), Sang phải (D) trong bộ giả lập Keyboard and Display MMIO Simulator).
* Vị trí bóng ban đầu ở giữa màn hình.
* Tốc độ bóng di chuyển là có thay đổi không đổi.
* Khi người dùng giữ một phím nào đó (W, S, A, D) thì quả bóng sẽ tăng tốc theo hướng đó với gia tốc tuỳ chọn.

1. Giải thích thuật toán
2. Bước 1: Khởi tạo các giá trị cho chương
   * Thiết lập (x, y) = (256, 256) là vị trí đầu tiên của quả bóng
   * R = 16 (bán kính quả bóng)
   * px = py = 0 (độ dịch chuyển của quả bóng)
   * Thời gian ngủ của chương trình là 30ms
3. Bước 2: Tạo ra 1 mảng lưu vị trí tương đối của các điểm nằm trên viền quả bóng với trung tâm của quả bóng bằng cách

* Lặp giá trị px từ 0 đến R.
* Ứng với mỗi giá trị px, ta sẽ tính được py =
* Từ đó ta lưu 8 vị trí tương đối của các điểm nằm trên đường tròn vào mảng bao gồm
* (px, py)
* (px, – py)
* (– px, py)
* (– px, – py)
* (py, px)
* (py, – px)
* (– py, px)
* (– py, – px)

A diagram of a circle with a circle and a circle with a circle with a circle with a circle with a circle with a circle with a circle with a circle with a circle with a circle with

Description automatically generated

1. Bước 3: Đọc input từ người dùng

* Nếu người dùng nhập “a”: thiết lập px = -1, py = 0 (sang trái).
* Nếu người dùng nhập “d”: thiết lập px = 1, py = 0 (sang phải).
* Nếu người dùng nhập “w”: thiết lập px = 0, py = -1 (lên trên).
* Nếu người dùng nhập “s”: thiết lập px = 0, py = 1 (xuông dưới).
* Ta sử dụng một thanh ghi để lưu lại phím mà người dùng ấn trước đó.
* Nếu người dùng giữ một phím thì quả bóng sẽ tăng tốc theo hướng đó.

1. Bước 4: Kiểm tra xem quả bóng có đập vào cạnh của màn hình hay không bằng cách:

* Kiểm tra hoành độ điểm bên phải (x + R) nhỏ hơn 512 không?

(Nếu lớn hơn hoặc bằng 512 là đập vào màn hình).

* Kiểm tra hoành độ điểm bên trái (x – R) lớn hơn 0 không?

(Nếu nhỏ hơn hoặc bằng 0 là đập vào màn hình).

* Kiểm tra tung độ điểm dưới cùng (y + R) nhỏ hơn 512 không?

(Nếu lớn hơn hoặc bằng 512 là đập vào màn hình).

* Kiểm tra tung độ điểm trên cùng (y – R) lớn hơn 0 không?

(Nếu nhỏ hơn hoặc bằng 0 là đập vào màn hình).

* Nếu quả bóng đập vào cạnh của màn hình, ta sẽ đảo ngược hướng của quả bóng, đồng thời đảo ngược giá trị của thanh ghi lưu phím người dùng vừa ấn.
* Nếu không, ta sẽ di chuyển quả bóng tới vị trí mới.

1. Bước 5: Di chuyển quả bóng tới vị trí mới

* Xóa quả bóng ở vị trí cũ bằng cách set giá trị màu sắc của mọi điểm thuộc viền của quả bóng thành màu đen.
* Cập nhật lại vị trí của tâm quả bóng tới tọa độ mới (x = x + px, y = y + py).
* Vẽ lại quả bóng ở vị trí mới bằng cách set giá trị màu sắc của mọi điểm thuộc viền của quả bóng thành màu vàng.
* Sau đó, ta quay lại bước 3.

1. Giải thích chương trình
2. Hàm “initialization”

A screenshot of a computer

Description automatically generated

* Hàm “initialization” khởi tạo các giá trị cho chương trình trong đó ($s0, $s1) = (x, y) lưu tọa độ tâm của quả bóng.
* $s2 (px), $s3 (py) là độ dịch chuyển của quả bóng so với vị trí hiện tại.
* Bán kính của quả bóng R = 16.
* Thời gian ngủ mặc định của chương trình $s4 = 30 ms.
* $t8 lưu kí tự mà người dùng đã nhấn trước đó.
* Mảng “circle\_point” là mảng lưu vị trí tương đối của các điểm nằm trên viền quả bóng với trung tâm của quả bóng.

1. Hàm “create\_circle\_point”

A white background with green text

Description automatically generated

A white background with green text

Description automatically generated



* Hàm “create\_circle\_point” thiết lập giá trị cho mảng circle\_point bằng cách:
* Lặp giá trị $a1 = px từ 0 đến R, tại mỗi giá trị px
* Ta tính $a2 = py =
* Sau đó lưu 8 giá trị sau vào trong mảng “circle\_point”
* (px, py)
* (px, – py)
* (– px, py)
* (– px, – py)
* (py, px)
* (py, – px)
* (– py, px)
* (– py, – px)
* Quá trình lặp sẽ dừng lại khi px = R.
* Hàm “sqrt” dùng để tính căn bậc 2 của .

1. Hàm “input”

A white background with black dots

Description automatically generated

* Hàm “input” nhận giá trị kí tự mà người dùng nhập.
  + Nếu người dùng nhập “a”: thiết lập $s2 (px) = -1, $s3 (py) = 0 (sang trái).
  + Nếu người dùng nhập “d”: thiết lập $s2 (px) = 1, $s3 (py) = 0 (sang phải).
  + Nếu người dùng nhập “w”: thiết lập $s2 (px) = 0, $s3 (py) = -1 (lên trên).
  + Nếu người dùng nhập “s”: thiết lập $s2 (px) = 0, $s3 (py) = 1 (xuông dưới).
  + Ta sử dụng một thanh ghi để lưu lại phím mà người dùng ấn trước đó.
  + Nếu người dùng giữ một phím thì quả bóng sẽ tăng tốc theo hướng đó.
* Khi nhận được input của người dùng, chương trình sẽ cập nhật độ dịch chuyển của tâm quả bóng, sau đó kiểm tra xem liệu quả bóng có đập vào cạnh của màn hình hay không.

A white background with black and white clouds

Description automatically generated

1. Hàm “check\_hit\_edge”

A screenshot of a computer

Description automatically generated

A white background with black dots

Description automatically generated

* Chương trình sẽ kiểm tra lần lượt điểm bên phải, bên trái, thấp nhất, cao nhất của quả bóng.
* Nếu quả bóng đập vào cạnh của màn hình, chương trình sẽ chuyển tới hàm “reverse” để đảo ngược hướng của quả bóng.
* Nếu không sẽ cập nhật lại vị trí của quả bóng trên màn hình Bitmap.
* Hàm “reverse” đảo ngược hướng di chuyển của quả bóng, đồng thời cập nhật lại giá trị của thanh ghi $t8.

1. Hàm “move\_cirlce”

A screenshot of a computer

Description automatically generated

A green text on a white background

Description automatically generated

* Quả bóng sẽ được di chuyển bằng cách:
* Xóa quả bóng ở vị trí cũ: chuyển tất cả các điểm trên viền của quả bóng thành màu đen.
* Cập nhật lại vị trí mới của tâm quả bóng: Địa chỉ của tâm quả bóng sẽ được tính bằng công thức 4 \* (512 \* x + y).
* Vẽ quả bóng ở vị trí mới: chuyển tất cả các điểm trên viền của quả bóng thành màu vàng.
* Sau đó chương trình sẽ chuyển về hàm “input” để lặp lại một chu kỳ mới.

1. Demo chương trình

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

s

Bài 2: Postscript CNC Marsbot

1. Yêu cầu bài toán

* Sử dụng Marsbot Tool trong phần mềm MARS MIPS để vẽ các từ DCE, SoICT và một hình dạng tự chọn (tối thiểu 10 đường cắt)
* Tạo postscript theo cấu trúc <góc chuyển động>, <thời gian>, <cắt/không cắt> được lưu trữ cố định bên trong mã nguồn để vẽ các từ nói trên
* Mã nguồn chứa 3 postscript và người dùng sử dụng 3 phím 0, 4, 8 trên bàn phím Key Matrix của Digital Lab Sim để chọn postscript được gia công:
  + Phím 0: chương trình sẽ thực thi postscript 0 (vẽ từ DCE)
  + Phím 4: chương trình sẽ thực thi postscript 4 (vẽ từ SoICT)
  + Phím 8: chương trình sẽ thực thi postscript 8 (vẽ hình trái tim)

1. Giải thích thuật toán

* **Bước 1:** **Khởi tạo**
  + Kích hoạt các interrupt cho Key Matrix
  + Bắt đầu vòng lặp vô hạn để chờ các ngắt từ bàn phím
* **Bước 2: Interrupt Service Routine**
  + Lưu trữ file thanh ghi hiện tại vào Stack
  + Xác định phím nào đã được bấm bằng cách kiểm tra từng hàng của Key Matrix
  + Dựa trên phím được nhấn (0, 4, 8) để nhảy đến các đoạn mã tương ứng để vẽ các từ theo yêu cầu của bài toán
* **Bước 3: Logic vẽ MarsBot Draw**
  + Logic vẽ được điều khiển bởi một script được lưu trữ trong bộ nhớ. Mỗi script bao gồm một chuỗi các hướng dẫn với ba tham số: góc, thời gian và cắt/không cắt.
  + Thuật toán xử lý từng bước di chuyển trong postscript
    - Góc: Xoay MarsBot đến góc chỉ định
    - Thời gian: Di chuyển MarsBot trong khoảng thời gian chỉ định
    - Cắt/Không cắt: Xác định liệu MarsBot có nên để lại dấu vết hay không
* **Bước 4: Các subroutine của MarsBot**
  + GO
  + STOP
  + TRACK
  + UNTRACK
  + ROTATE
* **Bước 5: Định nghĩa Postscript**
  + Postscript0: chuỗi các bước để vẽ từ “DCE”
  + Postscript4: chuỗi các bước để vẽ từ “SoICT”
  + Postscript8: chuỗi các bước để vẽ hình trái tim

1. Giải thích chương trình
2. Khai báo địa chỉ (address) cho MarsBot, Key Matrix và 3 postcript.

A screenshot of a computer

Description automatically generated

1. Trong chương trình chính, kích hoạt Interrupt của Key Matrix và tạo vòng lặp vô hạn để chờ interrupt.

A screenshot of a computer program

Description automatically generated

1. Lưu trữ ngữ cảnh của chương trình chính vào stack

A screenshot of a computer program

Description automatically generated

1. Chương trình get\_key giúp kiểm tra có phím nào được bấm hàng 1,2,3 và re-enable bit 7. Nếu phát hiện phím được bấm chương trình sẽ nhảy đến chương trình con key\_pressed

A screenshot of a computer program

Description automatically generated

1. Chương trình key\_pressed giúp kiểm tra xem phím nào được bấm trong 3 phím 0, 4, 8.

* Nếu phím được bấm trong 3 phím 0, 4, 8 thì ta gán địa chỉ bắt đầu của postscript tương ứng vào thanh ghi $a2 và địa chỉ kết thúc của postscript vào thanh ghi $a1 và chuyển tới chương trình con MarsBot\_Draw.
* Nếu phím được bấm không là 1 trong 3 phím 0, 4, 8 thì sẽ chuyển tới end\_script

A screenshot of a computer code

Description automatically generated

1. Chương trình MarsBot\_Draw có nhiệm vụ đọc phần postscript và vẽ theo postscript đã có sẵn.

A screenshot of a computer program

Description automatically generated

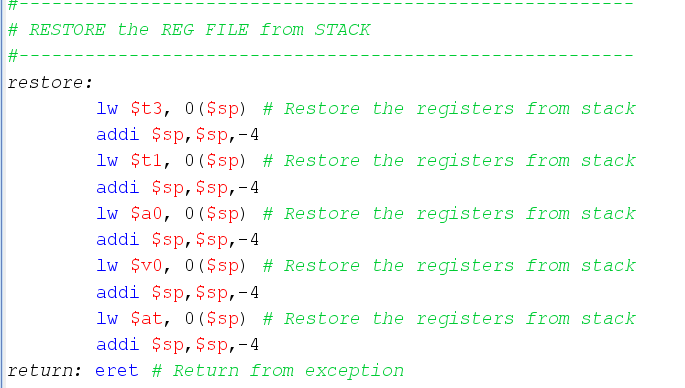
* Đầu tiên chương trình sẽ đọc góc mà con bot cần di chuyển từ phần tử đầu tiên sau đó giá trị góc đó sẽ được lưu vào thanh ghi vào thanh ghi $a0, sao đó nhảy đến chương trình ROTATE của Marsbot để xoay góc của con bot
* Sau đó phần tử tiếp theo được đọc là thời gian chuyển động của bot và sẽ được gán vào thanh ghi $a0
* Phần tử cuối cùng sẽ được đọc để xác đinh rằng bot có để lại vết (track) khi nó di chuyển hay không. Sau đó chuyển đến (nhảy) đến chương trình GO để con bot bắt đầu di chuyển theo góc đã và thời gian đã được xác định ở 2 chương trình trên.
* Cuối cùng, chương trình sẽ chuyển đến (nhảy) đến chương trình UNTRACK để dừng việc để lại vết và tiếp tục đọc các phần tử tiếp theo trong postscript cho đến khi nội dung ở thanh ghi $a2 trùng với nội dung của thanh ghi $a1 (địa chỉ kết thúc của postscript). Khi $a2 = $a1 thì chương trình chuyển đến end\_script để dừng con bot

1. Tăng địa chỉ chứa trong thanh ghi epc bằng cách sử dụng 2 lệnh mfc0 để đọc thanh ghi trong bộ đồng xử lý C0 và mtc0 để ghi giá trị vào thanh ghi trong bộ đồng xử lý C0

A white paper with green and red text

Description automatically generated

1. Khôi phục giá trị của các thanh ghi đã lưu từ Stack và lệnh eret giúp quay trở về thực hiện chương trình chính sau khi xử lý xong các đoạn mã Exception bằng cách gán nội dung thanh ghi PC bằng giá trị trong thanh ghi $14(epc). Khi đó thanh ghi $14 (epc) sẽ chứa địa chỉ kế tiếp của chương trình chính.



1. Các chức năng của MarsBot

A screenshot of a computer program

Description automatically generated

A computer code with text

Description automatically generated with medium confidence

1. Kết quả

* Khi bấm phím 0:

A screenshot of a computer

Description automatically generated

* Khi bấm phím 4:

A screenshot of a computer

Description automatically generated

* Khi bấm phím 8:

A screenshot of a computer

Description automatically generated