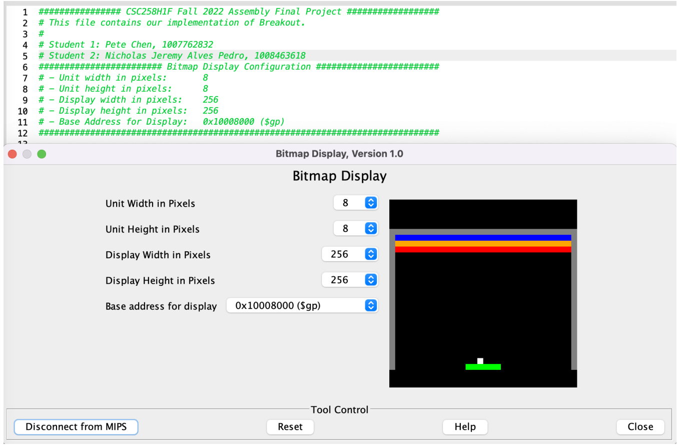
Project: Breakout Game

Pete Chen
Nicholas Jeremy Alves Pedro

November 28, 2022

1 Milestone 1-3

1. BitMap Display Configuration.



```
################ CSC258H1F Fall 2022 Assembly Final Project ##################
# This file contains our implementation of Breakout.
#
# Student 1: Pete Chen, 1007762832
# Student 2: Nicholas Jeremy Alves Pedro, 1008463618
######################## Bitmap Display Configuration ########################
# - Unit width in pixels:       8
# - Unit height in pixels:      8
# - Display width in pixels:    256
# - Display height in pixels:   256
# - Base Address for Display:   0x10008000 ($gp)
##############################################################################
```

Bitmap Display, Version 1.0

Bitmap Display

Unit Width in Pixels 8

Unit Height in Pixels 8

Display Width in Pixels 256

Display Height in Pixels 256

Base address for display 0x10008000 ($gp)

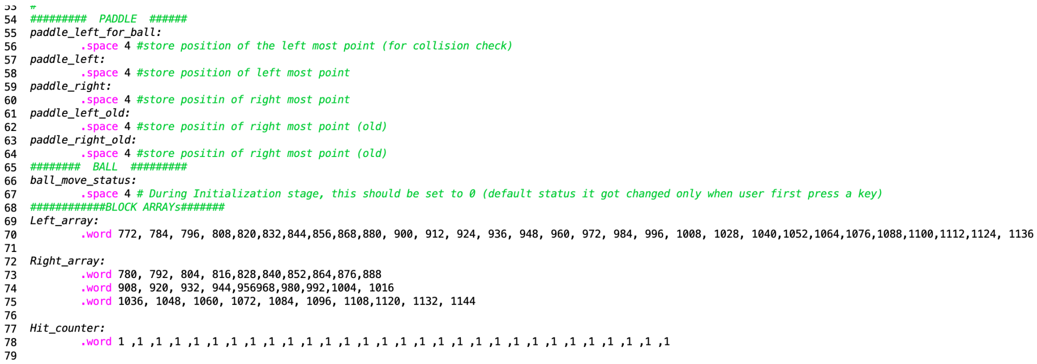
Figure 1: A picture of BitMap Display Configuration .

2. Plan of Memory.
   (a) Paddle: we decided to save the value of Left most and Right most positions of the paddle (both current and previous) in the memory. Since positions associated values are changed during the game, so we'll use data.space to save them, and mutate the value as game progresses.
   (b) Ball: we've decided to store Ball_move_status (which is used to determine whether the Ball should move or not) in memory. Since it will change (just once) during the game, we save it in data.space
   (c) Bricks: We saved both Left point location and Right point location of each brick in the memory. Since the setting of locations of bricks won't change during game progresses, we saved them in a series of data.word
   (d) Brick Hit_ Counter: We save the hit counter of bricks (which indicates which brick is destroyed and need to be erased) in memory in data.word as well (Hit counter contains 30 elements, even though they also change frequently during the game, but it is not possible to save them in preserved register, so we put them in memory)
   (e) NOTE: Other important values of Ball, e.g. ball direction, ball_x_coordinate, we saved them in Preserved Register instead of in memory because they will be mutated/changed very frequently so it's better to save them in preserved registers (convenient to access and change).
3. Plan of Collision.
   (a) We've decided different collision algorithms for wall collision, brick collision, and paddle collision
   (b) Wall Collision: When the ball collides with a wall, we check if the ball hit the side walls, if so, we "flip" the ball_x_speed (which indicates whether the ball overall movement is aligning to left or right) by multiplying

the current ball_x_speed by -1. When the ball collides with the top wall, we flips its ball_y_speed; If both happens, we flip both x and y speed (the case when the ball hits the top corners)

(c) Paddle Collision: Currently, the paddle has 6 spots (pixel), we assigned different angles to different spots: Left most = 25 degrees going left, left second spot = 45 degrees going left, left third spot = 75 degrees going left; the right three spots are symmetric but going right. For example, when the Ball hits the right most spot, no matter where the ball comes from (left, right, top), the ball will go 25 degrees to the right after collision.

(d) Brick Collision: If the ball hits a brick horizontally, we flip its X speed, if the ball hits a brick vertically, we flip its Y speed; If both happens at the same time, we flip both its X and Y speeds.



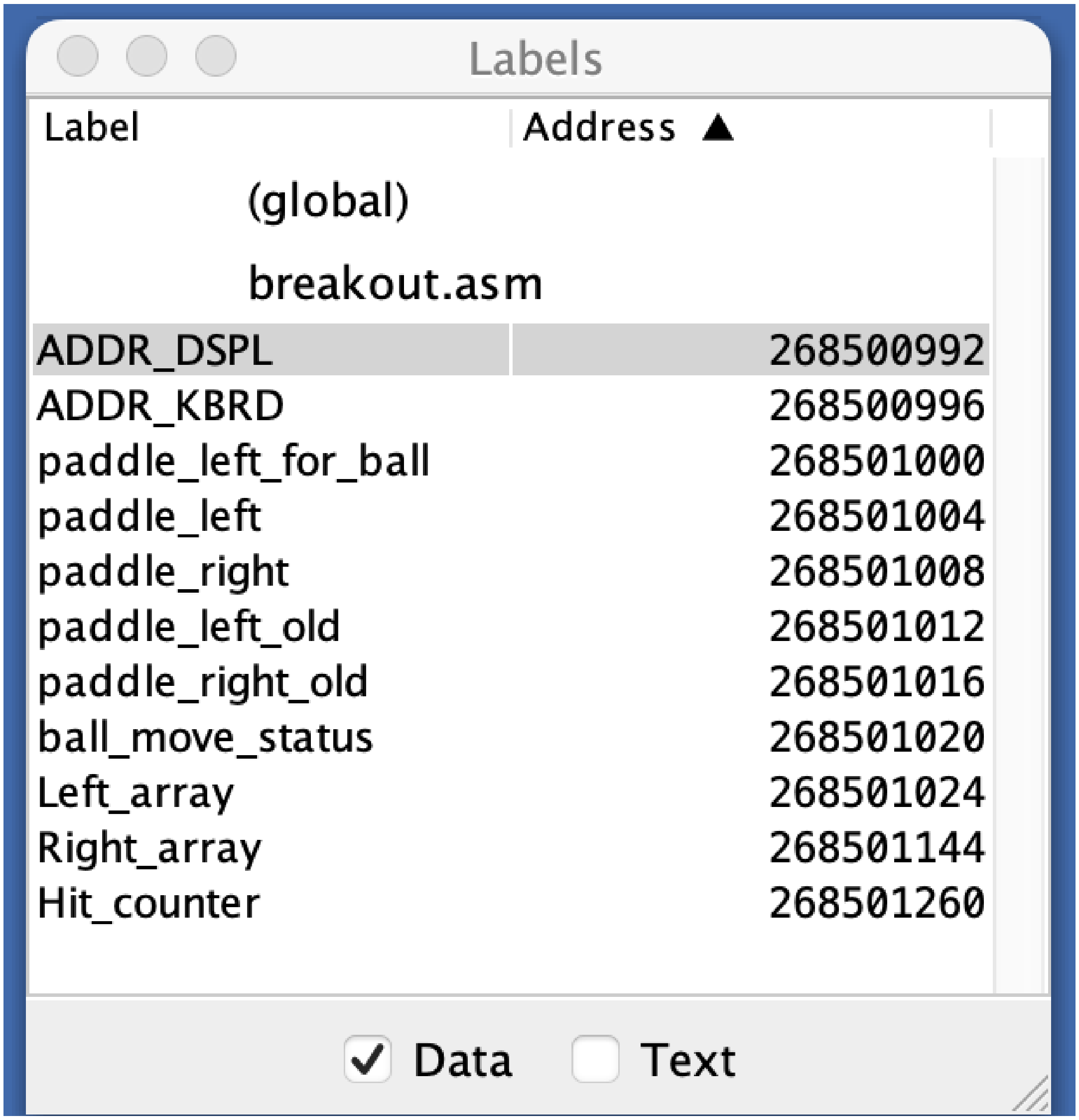
```
#
#########  PADDLE  ######
paddle_left_for_ball:
        .space 4 #store position of the left most point (for collision check)
paddle_left:
        .space 4 #store position of left most point
paddle_right:
        .space 4 #store positin of right most point
paddle_left_old:
        .space 4 #store positin of right most point (old)
paddle_right_old:
        .space 4 #store positin of right most point (old)
########  BALL  #########
ball_move_status:
        .space 4 # During Initialization stage, this should be set to 0 (default status it got changed only when user first press a key)
############BLOCK ARRAYs#######
Left_array:
        .word 772, 784, 796, 808,820,832,844,856,868,880, 900, 912, 924, 936, 948, 960, 972, 984, 996, 1008, 1028, 1040,1052,1064,1076,1088,1100,1112,1124, 1136

Right_array:
        .word 780, 792, 804, 816,828,840,852,864,876,888
        .word 908, 920, 932, 944,956968,980,992,1004, 1016
        .word 1036, 1048, 1060, 1072, 1084, 1096, 1108,1120, 1132, 1144

Hit_counter:
        .word 1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1

```

Figure 2: screenshots (or multiple screenshots) of memory



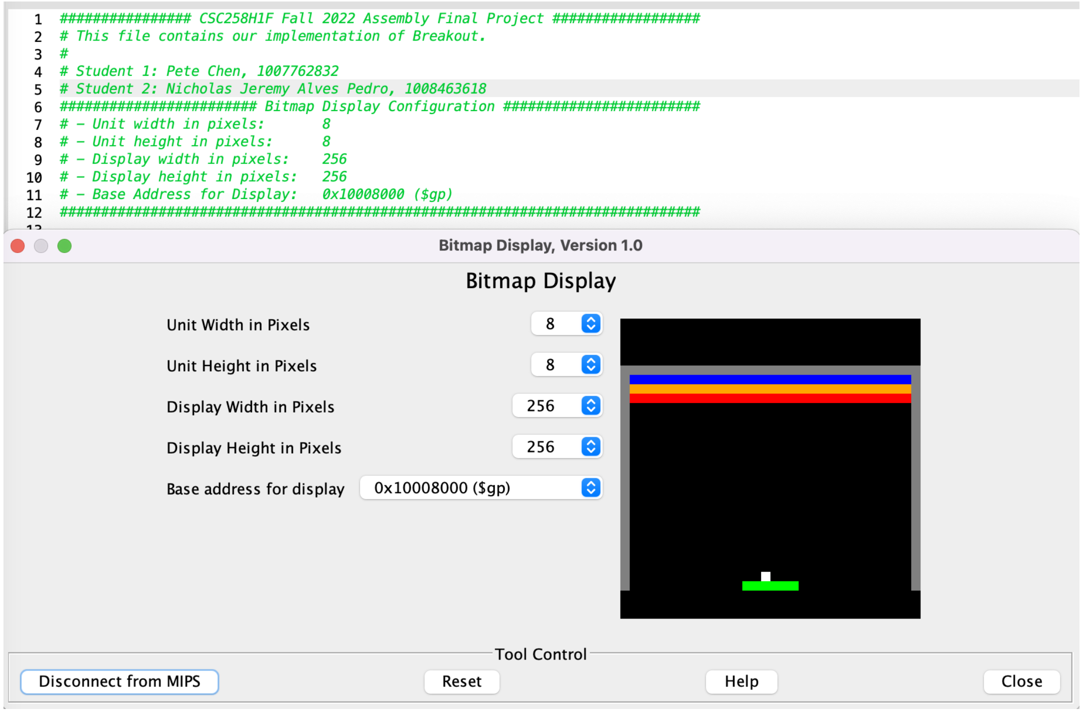
(global)

breakout.asm

Label	Address ▲
ADDR_DSPL	268500992
ADDR_KBRD	268500996
paddle_left_for_ball	268501000
paddle_left	268501004
paddle_right	268501008
paddle_left_old	268501012
paddle_right_old	268501016
ball_move_status	268501020
Left_array	268501024
Right_array	268501144
Hit_counter	268501260

☑ Data □ Text

Figure 3: screenshots (or multiple screenshots) of memory



```
################ CSC258H1F Fall 2022 Assembly Final Project ##################
# This file contains our implementation of Breakout.
#
# Student 1: Pete Chen, 1007762832
# Student 2: Nicholas Jeremy Alves Pedro, 1008463618
######################## Bitmap Display Configuration ########################
# - Unit width in pixels:       8
# - Unit height in pixels:      8
# - Display width in pixels:    256
# - Display height in pixels:   256
# - Base Address for Display:   0x10008000 ($gp)
##############################################################################
```

Bitmap Display, Version 1.0

Bitmap Display

Unit Width in Pixels 8

Unit Height in Pixels 8

Display Width in Pixels 256

Display Height in Pixels 256

Base address for display 0x10008000 ($gp)

Figure 4: A picture of Static Scene.