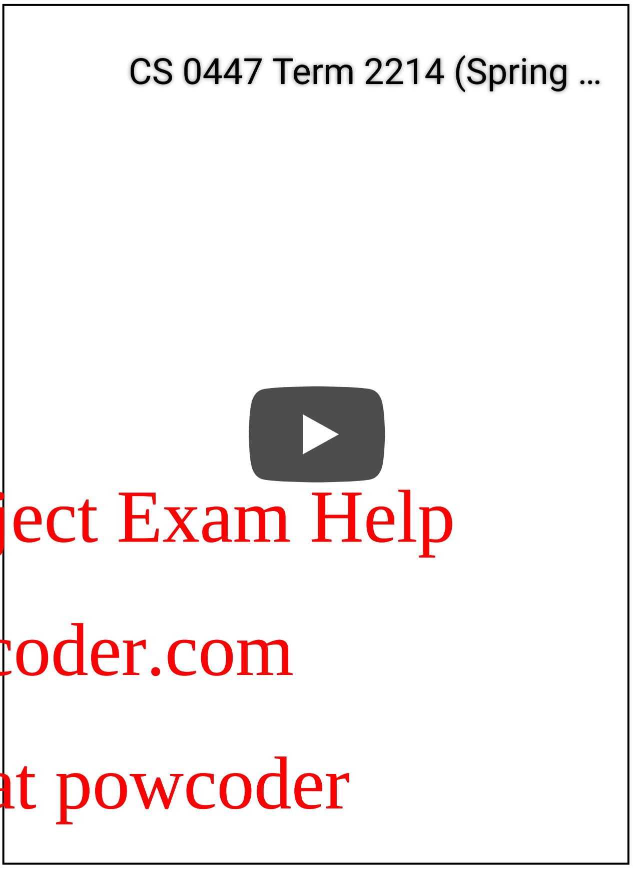
Project 1: Breakout!

Due by midnight, Saturday 3/13



For project 1, you'll be writing a video game in MIPS assembly: Breakout! If you're not familiar with the game, find an online version of it and play around. It's pretty simple - it was originally made *without a CPU* after all!

You'll continue to use the LED Keypad and Display plugin that you used in lab 4. To the right is a video demonstrating how the game will look when you're done (but you can make the blocks any pattern you like).

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Brief game description

In Breakout, you control the **paddle** at the bottom of the screen. You can move the paddle left and right.

The **blocks** are the colored rectangles at the top of the screen. Your goal is to **break all the blocks.** When all the blocks are broken, **the game ends.**

The way you do that is by **bouncing a ball** with your paddle. The ball breaks a block if it touches them, and then **bounces off the block.** The ball also bounces off **the walls and ceiling.**

If the ball goes off the bottom of the screen, that is a **miss,** andthe paddle and ball are reset (but the blocks remain).

When the game first starts, or whenever you miss the ball, the paddle should appear at a **random horizontal position** with the ball sitting on top of it. The ball will not move until **the player hits any key.** Then it will move up-right.

Grading Rubric

- **[10 points]:** Submission and style
    - **Please follow the submission instructions.**
    - **Code style is important.**
    - Instructions should be indented to the same level with the labels on the left.
        - *Indentation uses the tab key, not the spacebar. Never use the spacebar to indent.*
    - You should be following the calling conventions and register usage conventions.
    - You should also use multiple functions, where they make sense.
- **[90 points]:** The game
    - **[20]** Paddle
        - **[6]** Paddle appears at random X location
        - **[4]** Paddle is properly drawn
        - **[6]** Paddle moves left and right with arrow keys
        - **[4]** Paddle stops at edges of screen
    - **[46]** Ball
        - **[4]** Ball appears on top of paddle
        - **[4]** Ball is properly drawn
        - **[4]** Ball starts moving when player hits any key
        - **[4]** Ball moves!
        - **[4]** Ball bounces off walls
        - **[4]** Ball bounces off ceiling
        - **[8]** Ball bounces off paddle
        - **[8]** Ball bounces off all sides of blocks
        - **[6]** Ball going off the bottom of the screen resets paddle/ball and waits
    - **[24]** Blocks
        - **[8]** Blocks are properly drawn
        - **[4]** Count of remaining blocks to break is displayed onscreen
        - **[8]** Blocks disappear when hit by ball
        - **[4]** When all blocks are broken, program exits

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Stuff to download

Right-click and download this ZIP file. Your browser may automatically extract it to a folder. If not, open it and extract its contents to a **new folder.**

Now:

1. **Rename** `abc123_proj1.asm` to your username.
2. Open that file in MARS.
3. Open and connect the **Keypad and LED Display Simulator** tool.
4. **Assemble and run.** It should just sit there in an infinite loop.

What you’ve been given

- `abc123_proj1.asm` contains all the constants and variables you’ll need, as well as a skeleton `main` function.
    - **Read the comments.**
    - See the `# TODO` in a few places? That’s for you to do.
- `constants.asm` has the color and key constants for interacting with the display.
- `macros.asm` contains some very useful macros, which are like custom pseudo-instructions.
    - Each macro has a comment documenting what it does. You’re allowed to use any of them!
    - **I don’t recommend you make your own macros,** many people have gotten confused that way.
- `display_2211_0822.asm` is a library of functions to interact with the display and keypad.
    - This way you don’t have to deal with the MMIO directly, unlike in the lab.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

`enter` and `leave`

`count_blocks_left` looks like this:

```
count_blocks_left:
enter
    # TODO: actually implement this!
    li v0, 1
leave
```

What are `enter` and `leave`? They’re **macros** I provided in `macros.asm`:

- `enter` = `push ra`
- `leave` = `pop ra` and `jr ra`

So they act as the "braces" around functions.

If you want to use any `s` registers, you list them **in the same order** after `enter` and `leave`:

```
my_function:
enter s0, s1
    # in this function, I can use s0 and s1 as "local varia

# IMPORTANT: always list the same registers in the SAME ORD
leave s0, s1
```

These macros *greatly* reduce the amount of code you have to write for function prologues/epilogues. (I didn't give them to you until now because you had to learn how to do it correctly. Sorry, that's how it goes!)

Getting started, finally

Assignment Project Exam Help

Only change `abc123_proj1.asm`. Don't change the other 3 files.

https://powcoder.com

There are three main parts of this game, in order of increasing complexity: the **paddle,** the **blocks,** and the **ball.**

Add WeChat powcoder

The paddle

- In `main`, uncomment `# jal setup_paddle` and `# jal play_game`, then **stub those two functions out.**
- In `setup_paddle`, you need to do the equivalent of:

  ```
    paddle_x = rand(PADDLE_MAX_X - PADDLE_MIN_X) + PADDLE
  ```

  ○ **Syscall 42** gives a random value, but its arguments are weird:
    - ▪ always pass 0 in `a0`
    - ▪ pass the upper range (`PADDLE_MAX_X - PADDLE_MIN_X`) in `a1`

  ○ **Step through this function after implementing it** and make sure `paddle_x` is set to a random value.
- `play_game` should do the equivalent of:

```
// this is the game loop
do {
    draw_paddle();
    display_update_and_clear(); // from display_2211_
    wait_for_next_frame();      // also from display_
} while(count_blocks_left() != 0)
```

Tangent: using the drawing functions

There are a couple functions from `display_2211_0822.asm` you'll use to draw things to the screen:

- `display_set_pixel(x: a0, y: a1, color: a2)`
  - sets the pixel at `(x, y)` to `color`.
  - **this will crash if you give it invalid coordinates, as a debugging feature.**
    - If your program crashes on a `tlti` or `tgei` instruction, you passed invalid coordinates!
- `display_fill_rect(x: a0, y: a1, width: a2, height: a3, color: v1)`
  - I was BAD and I made it take an argument in a `v` register! GASP! sue me
  - starting at the top-left corner `(x, y)`, fills a rectangle `width` pixels wide and `height` pixels tall with `color`
  - **again this crashes if you give invalid x/y coordinates.**

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Drawing the paddle

`draw_paddle` needs to do:

```
// remember the color is passed in v1, cause I'm a rebel
display_fill_rect(paddle_x, PADDLE_Y, PADDLE_WIDTH, PADDLE_
```

Once implemented, **the paddle should appear onscreen at the random location chosen by `setup_paddle`!**

Moving the paddle

Remember the lab with the dots? And how you used the arrow keys to move them? You'll be doing something very similar here…

- In your `play_game` function's loop, add a call to `check_input` and stub that out.
- In `check_input`, call `input_get_keys_held` (from `display_2211_0822`).
    - It returns the currently-held keys as a bitfield in `v0`, just like on the lab.
- The logic works something like this:
    - **If the player is holding `KEY_L`,** decrement `paddle_x`.
    - **If the player is holding `KEY_R`,** increment `paddle_x`.
    - `paddle_x` should never go less than `PADDLE_MIN_X` or greater than `PADDLE_MAX_X`.
        - You can use conditionals for this (no easy `and` this time)
        - Or you could check out the `min/max` macros...

Once implemented, **the paddle should move when you hit the left/right arrow keys!** Make sure it stops at the sides of the screen.

Assignment Project Exam Help

The blocks

https://powcoder.com

- The blocks are held in the `blocks` array. It is an array of **bytes.**
    - 0 means an empty space.
    - Anything other than 0 is the color of that block. (See the colors in `constants.asm`)

Add WeChat powcoder

- There are at most `BOARD_MAX_BLOCKS` blocks on-screen.
    - There are `BOARD_BLOCK_WIDTH` columns of blocks and `BOARD_BLOCK_HEIGHT` rows.
    - Each block is `BLOCK_WIDTH` pixels wide and `BLOCK_HEIGHT` pixels tall.
    - The bottom of the last row of blocks is `BOARD_BLOCK_BOTTOM` pixels down from the top.
- **Drawing the blocks:**
    - Your `play_game` loop also needs to call a `draw_blocks` function...
    - And `draw_blocks` needs to draw all the blocks in the array.
        - It's a 2D array, meaning a nested `for` loop.
        - And you'll be calling `display_fill_rect` inside the loops, meaning you'll need `s` registers for your loop counters...
        - You can figure it out. :)
    - The default `blocks` configuration is just 8 colored blocks in the middle of the screen...

        - You can change that array to whatever you want.

- **Showing how many blocks are left:**
    - There is a function `display_draw_int(x: a0, y: a1, value: a2)` that does what it says.
    - In your `play_game` loop, you should call a `show_blocks_left` function which does:

```
display_draw_int(3, 57, count_blocks_left())
```

    - But `count_blocks_left` is just returning 1 right now, so let's implement `count_blocks_left`.
- **Counting the blocks:**
    - In `count_blocks_left`, delete the `li v0, 1` line and replace it with code that does the following:

```
v0 = 0;
for(i = 0; i < BOARD_MAX_BLOCKS; i++) {
    if(blocks[i] != 0) {
        v0++
    }
}
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

    - Tips:
        - You don't actually need an `s` register for `i`, because you never call a function in the loop.
        - Remember to use the *name* of the constant, not its value, when writing the condition.
        - If you haven't encountered this load/store syntax yet:

```
    lb t0, blocks(t1)
```

it means, "load a byte from `blocks + t1` into `t0`".

          - So, it does the `la` and `add` steps for you.
          - And since `blocks` is an array of bytes, there is no `mul` needed to calculate `Si`.
          - Do you think I wrote it this way on purpose? To make it easier for you? We may never know...

The Ball

Okay, you’re mostly on your own for this. By now you should have a good idea where to put code to implement the various behaviors of the ball.

- The ball is drawn as a single pixel using `display_set_pixel` .
- The ball’s **velocity** (`ball_vx/ball_vy`) is how far it moves each time through the loop.
    - It will always be diagonal, so (1, 1), (1, -1), (-1, 1), or (-1, -1).
- After setting up the paddle, set up the ball:
    - It should appear on top of the paddle. Figure out the coordinates for that.
    - Set its velocity to move up and to the right.
    - **It should not start moving until the player presses a key.**
        - So when the game starts, it should show everything but the ball should be frozen in place.
        - Then, the player hits a key (arrow key or zxcb) and it begins to move.

Assignment Project Exam Help

- Moving the ball means **adding the velocity to its position.**
    - It’s as simple as `ball_x += ball_vx` and same for y.

https://powcoder.com

- But of course, the ball wants to go offscreen and crash your program… what a PAIN

Add WeChat powcoder

Collision

The `ball_old_x/y` variables are used for collision. You use them like so:

1. set `ball_old_x = ball_x` and `ball_old_y = ball_y`
2. do `ball_x += ball_vx` to move it on the X, and then…
    1. check if the ball ran into anything
    2. if it did, `ball_x = ball_old_x` to back up to the last valid X, and **negate** `ball_vx`
3. `ball_y += ball_vy`
    1. check if the ball ran into anything
    2. if it did, `ball_y = ball_old_y` to back up to the last valid Y, and **negate** `ball_vy`

The tricky bits in the above are the “check if the ball ran into anything”.

- **Walls and ceiling**

    - When the ball hits a wall or the ceiling, it just bounces off as described above.
- **The paddle**
    - The ball hits the paddle if `ball_y == PADDLE_Y` *and* `ball_x >= paddle_x` *and* `ball_x < paddle_x + PADDLE_WIDTH`.
        - The only way this can happen is during the "moving on the Y" step, so this can only bounce it upwards.
- **The bottom of the screen**
    - When the ball hits the bottom of the screen, `off_screen` should be set to 1.
    - Change your `play_game` loop to break if `off_screen` is not 0.
    - This will cause `play_game` to return to `main`, which will re-set-up the paddle and ball.
        - You will then have to set `off_screen` back to 0 when you setup the ball.
    - **Make sure it waits for the player to hit a key before moving the ball again!**
- **The blocks**
    - Well…

Assignment Project Exam Help

https://powcoder.com

Colliding with (and breaking) blocks

Add WeChat powcoder

This seems intimidating, but it is surprisingly simple. Here are some hints:

- This can be done in **constant time.** That means you don't need any loops.
- It **doesn't matter what direction the ball is moving.** `ball_vx/vy` are not needed.
- The blocks are stored in a 2D array. The ball has 2D coordinates.
    - That means there is a way to map from the ball's coordinates to array indexes.
- `BOARD_BLOCK_BOTTOM` is an important thing to consider, unless you like accessing past the end of the array.
    - If you like that, shame on you!!
- "Breaking" the block just means **storing a zero into that array element.**
    - `draw_blocks` will automatically not-draw it on the next frame.
    - And `count_blocks_left` will automatically count fewer blocks, meaning the number onscreen goes down.
    - Isn't code modularity *nice?*
- Maybe you should put all this logic into a function that **returns a boolean saying whether or not a block was broken,** so that you can **call this**

function in two places
  - Once for moving the ball on the X, once for moving it on the Y.
  - And that way, it will bounce in the correct direction, too.
  - Isn't code reuse *nice?*

Once this is working, you're... done! When the number of blocks remaining hits 0, the program should exit (since both `play_game` and `main`'s loops break when the count of blocks hits 0).

Submission

Be sure to review the grading rubric before submitting.

Assignment Project Exam Help

You will submit a ZIP file containing:

- Your `abc123_proj1.asm` file (but renamed with your username).
  - *Put your name and username at the top of the file in comments.*
  - **Also put any important notes to the TA at the top of this file in comments.**
    - For example, if you wrote some code that is never called, they will not see the behavior; tell them that you attempted it and you may get some partial credit.
- **All** the other `.asm` files I gave you.

https://powcoder.com

Add WeChat powcoder

The TA should be able to unzip your ZIP file, open your `_proj1.asm` file in MARS, and assemble and run it without a problem.

To make a ZIP file:

1. In your file browser, select all the files you want to add to the ZIP file (the files listed above).
2. **Right click on the files,** and...
   - **Windows:** do **Send To > Compressed (zipped) folder**. Then you can rename it.
   - **macOS:** do **Compress *n* items**. Then you can rename the `Archive.zip` file.

- **Linux:** I'm sure you already know.

Then, once you've made the ZIP file, **make sure to name it correctly.** My username is `jfb42`, so:

- ✅ `jfb42_proj1.zip` - the one and only acceptable filename.
- ❌ `jfb42_proj1` - no extension
- ❌ `JFB42_proj1.zip` - uppercase is bad
- ❌ `jfb_proj1.zip` - incomplete username
- ❌ `proj1.zip` - no username
- ❌ `jfb42_project1.zip` - it's `proj1`, not `project1`
- ❌ `jfb42_proj01.zip` - it's `proj1`, not `proj01`
- ❌ literally anything other than the first thing on this list

Submit here. Drag your asm file into your browser to upload. **If you can see your file in the folder, you uploaded it correctly!**

You can also re-upload to resubmit. It will overwrite your old submission (but we can still access the old one through Box).

Assignment Project Exam Help

https://powcoder.com

© 2016-2021 Jarrett Billingsley

Add WeChat powcoder