Instructions

To run:

1. Load the "super-mario.asm" file into MARS (this may take 15-20 seconds for it to load)
2. Then hit "assemble" (again, will take 15-20 seconds)
3. Open the Bitmap Display with the following settings:
    1. unit pixel width = 2
    2. unit pixel height = 2
    3. width = 1024
    4. height = 512
    5. source = $gp (global pointer
4. Open the keyboard simulator and full-screen the Bitmap Display
5. Hit "Run" and then "connect to mips" on both the bitmap and keyboard display
6. Use the WASD keys to move the player around the screen. Walking off the screen wraps you around the start.

Written Overview

I was very interested in creating animations in assembly, so I chose to create an animated mario character for my MIPS project. Due to time constraints, I stopped there and didn't create an actual playable game, but this code could easily be refactored and continued to represent a super mario level.

Since I knew before starting that the codebase for this project would become progressively messier and more difficult to debug, I decided to create macros for most small-but-annoying tasks in order to make later development more painless. This included single line drop-in replacements for

1. looping over a range and invoking another macro for each iteration step
    1. variants for reverse for loops and with step-sizes
2. loading in immediate values for registers and floats
3. converting between and doing arithmetic with int and float values
4. printing any int, float, string
5. bitshifting
6. sleep (delay)

I also had compound macros that invoked other macros, such as:

1. logging all player data
2. computing player distance from ground
3. drawing a single pixel at a certain x, y coordinate
    1. variants with different argument types (immediate vs register)
4. get memory address for a certain x, y location for a pixel in the bitmap
5. overlay black pixels on the old mario character for frame updates
6. increment/decrement player velocity x/y
7. increment/decrement player position x/y
8. processing input character
9. drawing mario run/idle frames 0-7

You can find a flowchart of the most high-level macros and how they're used in the flowchart diagram below.

Using macros was a critical part of development since it made debugging code significantly easier.

Now let's do a high-level walkthrough of the codebase and talk about some important design decisions.

Register and Memory Usage

- $v0 is strictly used for system calls
- $v1 is used for return values
- $a0 - $a3 are used for passing arguments to macros
- $t0 - $t9 are used for temporary values
- $s0 - $s7 are used for persistent variables (X, Y, color, etc.)
- $s0 is player X coordinate
- $s1 is player Y coordinate
- $s2 is reserved for color
- $s4 is reserved for direction --> -1 is left, 1 is right
- $s7 is the frame counter for mario's animation
  For all loops:
- $t0 = start
- $t1 = end
- $t2 = step size (if applicable)
- $t3 = iterator

Graphics Macros

I wrote python code (which is also supplied in this submission) that was used to convert the following 2 gifs of mario running and idling (standing) into individual frames, then process those into lines of code in MIPS that use my draw_pixel macro.

This gave thousands of commands using the macro `draw_pixel_with_color_and_offset_immediate(x, y, color)`

ex.
`draw_pixel_with_color_and_offset_immediate(24, 0, 0x007D2F3A)`

Main Program Flow

Here's some of the higher-level labels

main:

- initializes player variables (position x/y and velocity x/y) and frame counter.

main_loop:

- sleeps ever 33ms
- then logs data
- processes input, and updates player if necessary

update_player:

- updates physics 3x (3 time-steps)
- re-renders player if difference in distance exceeds a certain threshold constant

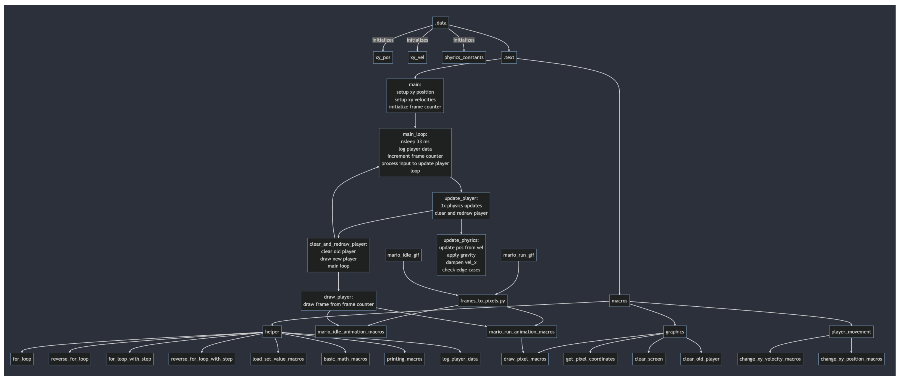
update_physics:

- updates position from velocity
- applies gravity to velocity
- dampens x velocity if on ground
- snaps velocity to 0 if it's too low <0.5

clear_and_redraw_player:

- clears old player pixels and draws new location

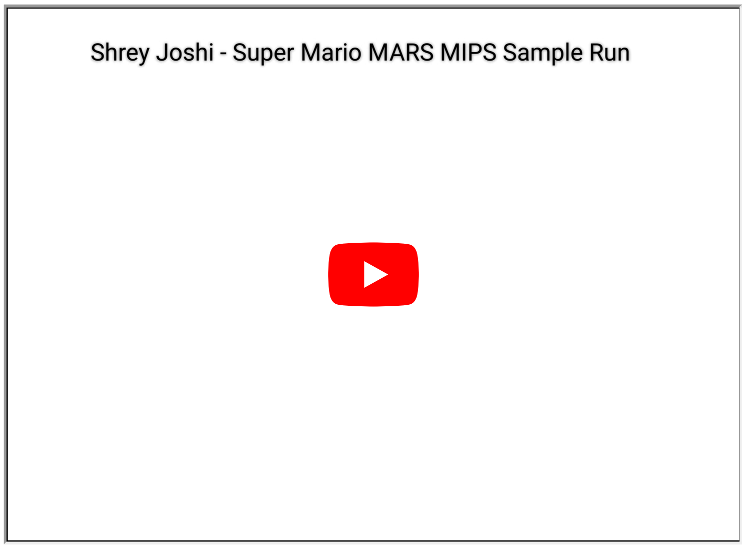
Flowchart

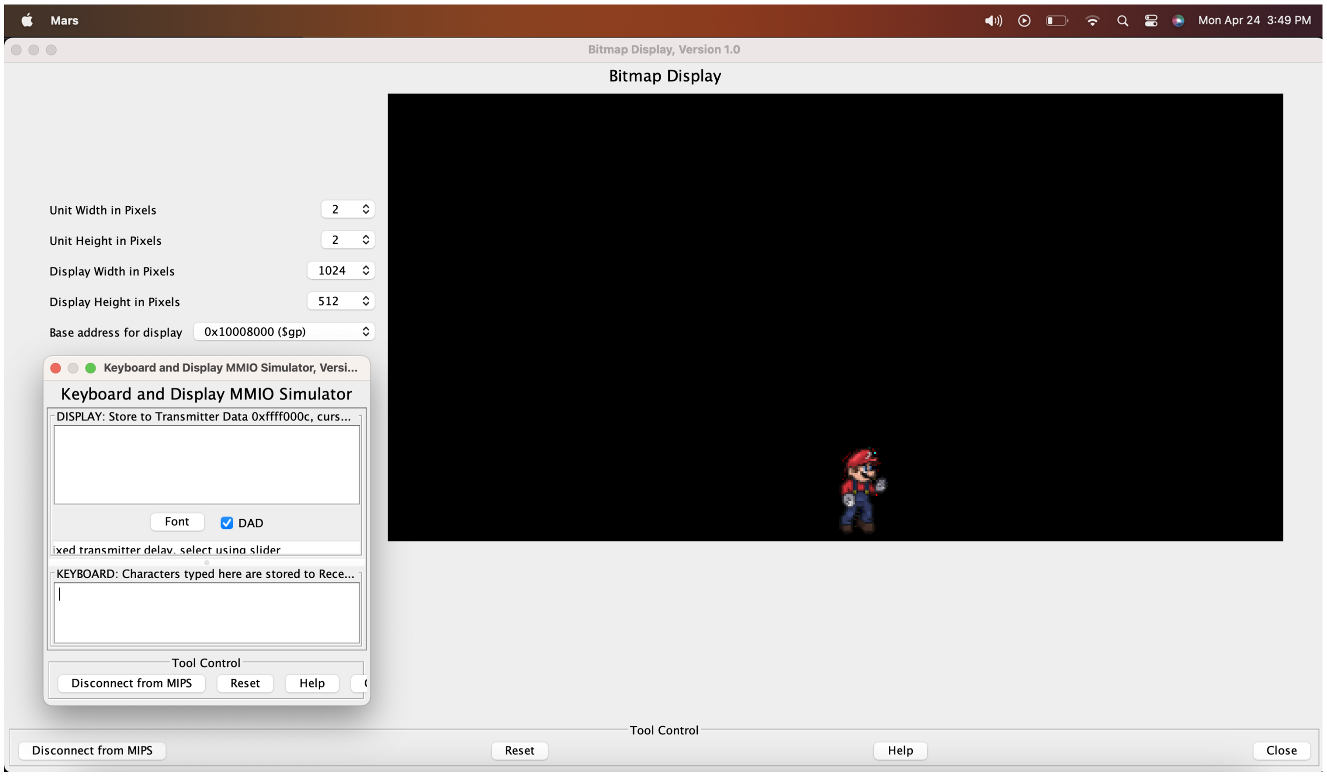


See flowchart with zoomable interactive interface here:
https://mermaid.live/view#pako:eNqFVU1v2zAM_SuCzmmQrzqJB-zU607bafMgaDabCLUlQZLTuF3_-yjJju3UyU6RyUfykXyK3mmuCqApPRiuj-
TH05dMZrLiRihmaskO4pk8PHwlz4ZXYJlTTIszlHaumw4mihLu4zI5YQ7wvhCXouJOKMkqnhtl_xcTqn4OyuS84I4H4F8hhRO8FG9g_5Jzw7RH3PGfoLzt18fGityyXEnr
bYR0Qh6mvaU6sLZ3P1vv7eYT_IXhr1EQg_gR4oDkIiBXyhRCcgefUXkJ3DCbGwB5w6nKoqXiAVcLibgjlwdgUToqF665kLoPRyWKa90OdCPkr4z6nzTLJA671ihP0gU
jioU8lIgbLmtEs6-lLQlgCbrNal8PtxFq0MSduEr4MBCP-MC2K9ROVhLhNS1I06RWmMMtPEhGSqyJxJqBjYR2E4aGY2-Pav1ubt8LdZzCysiXBYoXa-
JrlCsMMoxqhITDcpEg6_TMcYb-
GxU5WeLVq512fg7esL14nfBKw3SO9nZ8zhC_kKgOOAwuEVqtwhETSFhFgn3Dd_wpJcuUYj9HEOzEl57S9gnGcz3RsLx7u_DxgyvWAUKUQBhUFNi-
1z3zr_2NHj6WRi-Ttfe6X8IOqMVGGy8wHfuHedFMuqOKOOMpngsuHnJaCY_EMdrp743MqepMzXMWpE8CY77rzojFMIp8y2-m-H5nFHN5U-
lLhD8pOk7PdP0cb5fPO42y2SfbDe7x91yRhuaLjfzxWa7W21W6-1inSyT7ceMvoUEi_l-v94nu_Vqsdkni90q-fgHbwzR7A

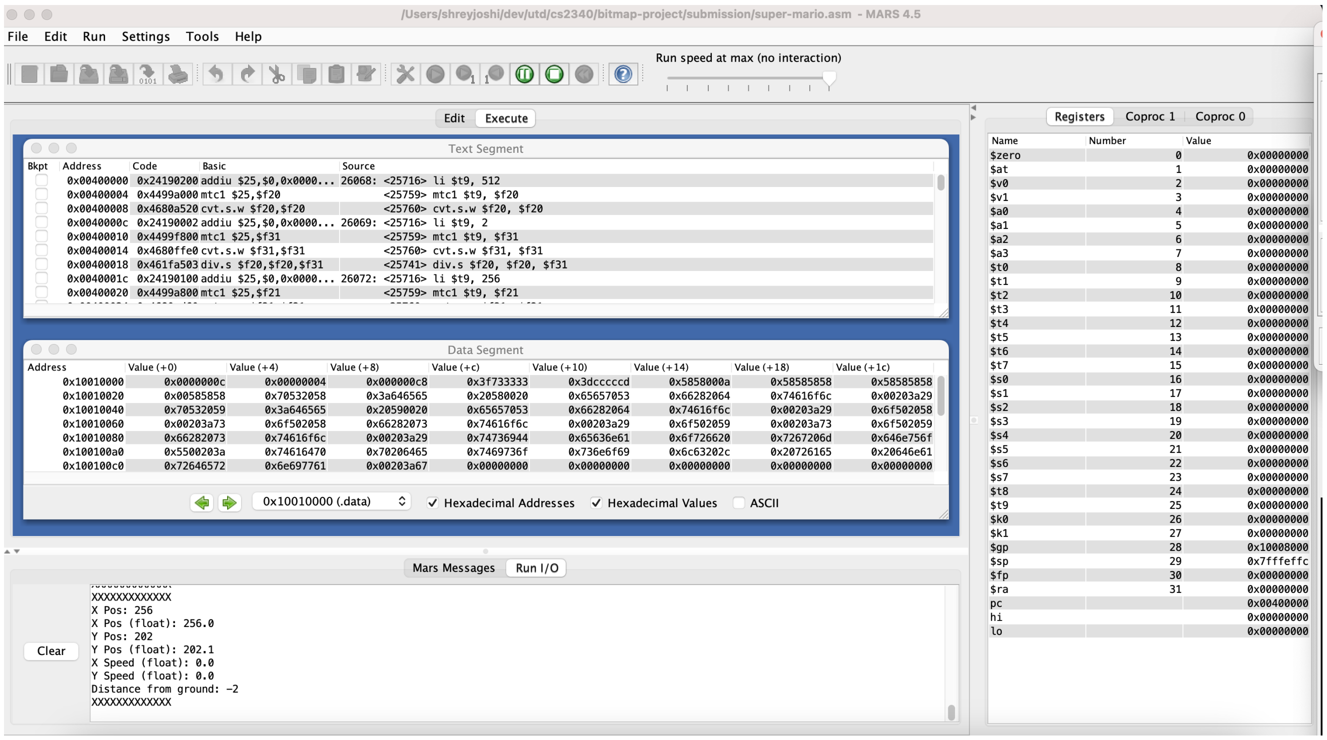
Screenshots and Sample run video

Sample run: https://youtu.be/LhNIEds_Tz0




Mars
Mon Apr 24 3:49 PM
Bitmap Display, Version 1.0
Bitmap Display
Keyboard and Display MMIO Simulator, Versi...
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c, curs...
Font
DAD
ixed transmitter delay, select using slider
KEYBOARD: Characters typed here are stored to Rece...
Tool Control
Disconnect from MIPS
Reset
Help
Tool Control
Disconnect from MIPS
Reset
Help
Close

Unit Width in Pixels 2
Unit Height in Pixels 2
Display Width in Pixels 1024
Display Height in Pixels 512
Base address for display 0x10008000 ($gp)


/Users/shreyjoshi/dev/utd/cs2340/bitmap-project/submission/super-mario.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Data Segment
0x10010000 (.data)
Registers Coproc 1 Coproc 0
Mars Messages Run I/O
Clear

Bkpt	Address	Code	Basic		Source
	0x00400000	0x24190200	addiu $25,$0,0x0000...	26068:	<25716> li $t9, 512
	0x00400004	0x4499a000	mtc1 $25,$f20		<25759> mtc1 $t9, $f20
	0x00400008	0x4680a520	cvt.s.w $f20,$f20		<25760> cvt.s.w $f20, $f20
	0x0040000c	0x24190002	addiu $25,$0,0x0000...	26069:	<25716> li $t9, 2
	0x00400010	0x4499f800	mtc1 $25,$f31		<25759> mtc1 $t9, $f31
	0x00400014	0x4680ffe0	cvt.s.w $f31,$f31		<25760> cvt.s.w $f31, $f31
	0x00400018	0x461fa503	div.s $f20,$f20,$f31		<25741> div.s $f20, $f20, $f31
	0x0040001c	0x24190100	addiu $25,$0,0x0000...	26072:	<25716> li $t9, 256
	0x00400020	0x4499a800	mtc1 $25,$f21		<25759> mtc1 $t9, $f21

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x0000000c	0x00000004	0x000000c8	0x3f733333	0x3dcccccd	0x5858000a	0x58585858	0x58585858
0x10010020	0x00585858	0x70532058	0x3a646565	0x20580020	0x65657053	0x66282064	0x74616f6c	0x00203a29
0x10010040	0x70532059	0x3a646565	0x20590020	0x65657053	0x66282064	0x74616f6c	0x00203a29	0x6f502058
0x10010060	0x00203a73	0x6f502058	0x66282073	0x74616f6c	0x00203a29	0x6f502059	0x00203a73	0x6f502059
0x10010080	0x66282073	0x74616f6c	0x00203a29	0x74736944	0x65636e61	0x6f726620	0x7267206d	0x646e756f
0x100100a0	0x5500203a	0x74616470	0x70206465	0x7469736f	0x736e6f69	0x6c63202c	0x20726165	0x20646e61
0x100100c0	0x72646572	0x6e697761	0x00203a67	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

✓ Hexadecimal Addresses ✓ Hexadecimal Values ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

```
XXXXXXXXXXXXX
X Pos: 256
X Pos (float): 256.0
Y Pos: 202
Y Pos (float): 202.1
X Speed (float): 0.0
Y Speed (float): 0.0
Distance from ground: -2
XXXXXXXXXXXXX
```

Hints, Warnings, and Future Work

1. The code for animations consists of over 25,000 lines of calls to the draw_pixel macro with different coordinate and color arguments. This is obviously not the best way to do animations - a far more efficient way that I tried was converting the frame data into a different file format, then loading that into MIPS memory, but I kept running into memory errors so I ditched the approach for the easier way of just writing out all the function calls for every pixel for each frame.

2. Currently, when mario exits the screen, there is no code that puts his x position on the other side of the screen - instead what happens is that the x value is incremented beyond the max x value, so the y value effectively increases by 1 in the code that finds the memory address for the reference pixel.

3. The x velocity is dampened by 1% each frame if the player is on the ground, simulating friction, but this restriction does not apply when the player is in the air. Thus, it is possible to achieve infinite speed by repeatedly jumping and moving in one direction, since the screen wraps the player around whenever mario exits leaves the viewable area. This could be prevented by enforcing a maximum X velocity for ther player.