ARM Project

Group 12

Giacomo Guerci
Frances Tibble
Oliver Brown
Dylan Gape

Designing the Emulator

```
/**
 * @brief ARMMachine object
 * Object representing the state and functions of an ARM machine
 */
typedef struct {
    /** Array of 32 bit registers */
   uint32_t registers[NUMBER_REGISTERS];
    /** System memory */
   uint8_t *memory;

   uint32_t (*rd32)(void *self, uint32_t address);
   uint32_t (*rd32_le)(void *self, uint32_t address);
   void (*str32)(void *self, uint32_t address, uint32_t value);
   void (*destroy)(void *self);
   void (*print)(void *self);
} ARMMachine;
```

ARM Machine struct

```c
/** If ERR_LN is set print line numbers with warnings */
#ifdef ERR_LN
#define log_err(M, ...)  fprintf(stderr, "[ERROR] (%s:%d) " M "\n", __FILE__, __LINE__, __VA_ARGS__)
#define log_warn(M, ...) fprintf(stderr, "[WARN] (%s:%d) " M "\n", __FILE__, __LINE__, __VA_ARGS__)
#define log_info(M, ...) fprintf(stderr, "[INFO] (%s:%d) " M "\n", __FILE__, __LINE__, __VA_ARGS__)
#else
#define log_err(M, ...)  fprintf(stderr, "[ERROR] " M "\n", __VA_ARGS__)
#define log_warn(M, ...) fprintf(stderr, "[WARN] " M "\n", __VA_ARGS__)
#define log_info(M, ...) fprintf(stderr, "[INFO] " M "\n", __VA_ARGS__)
#endif

/** If A is false emit given message and skip to error handling label */
#define check(A, M, ...) if(!(A)) { log_err(M, __VA_ARGS__); goto error; }
```

```c
check_is_register(tokens);
check(tokens->val.operand == R0, "%s", "Expected r0");
```

```c
error:
    return UNDEF_INSTR;
```

Error Handling

Designing the Assembler

```
typedef struct {
    /** Symbol table mapping strings to their addresses */
    Hashmap *sym_tab;
    /** Table of forward references symbols to forward_ref */
    Multimap *fr_tab;
    /** Literal stack with fixups */
    literal_pool *lt_pool;
    /** Binary file to which all the instructions are saved */
    bin_file *file;
    /** Current address through binary */
    uint32_t addr_ctr;
    /** Memory offset for relocation */
    uint32_t addr_offset;
} asm_data;
```

Data Structures

Data Structures

Literal Pool

- Label literals (char *) and constant literals (uint32_t) to a dequeue of fixup_requests
- Constants and labels from ldr instructions

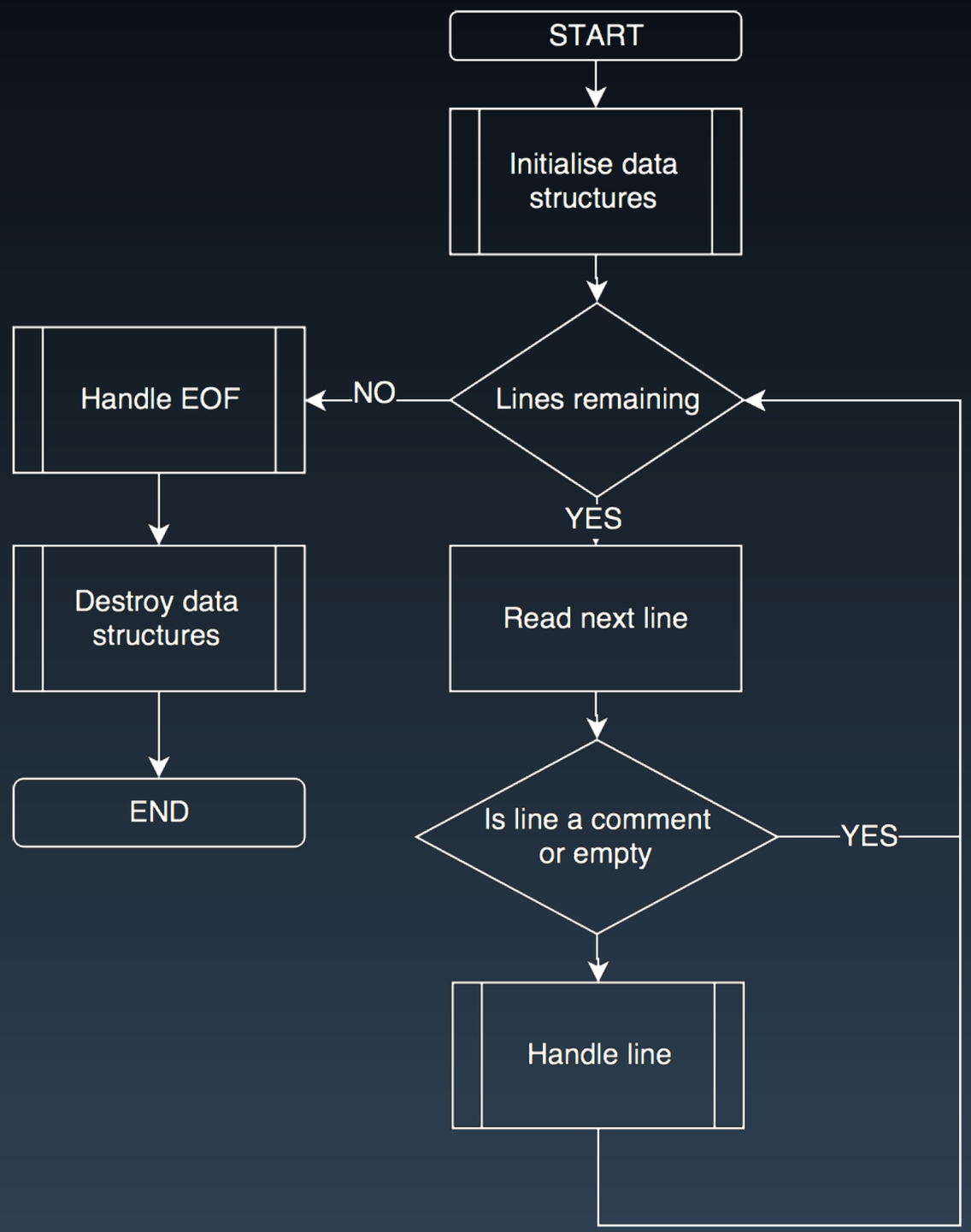
Forward Reference Table

- Labels (char *) to a dequeue of fixup_requests
- Labels we have seen in branch statements but not actually defined

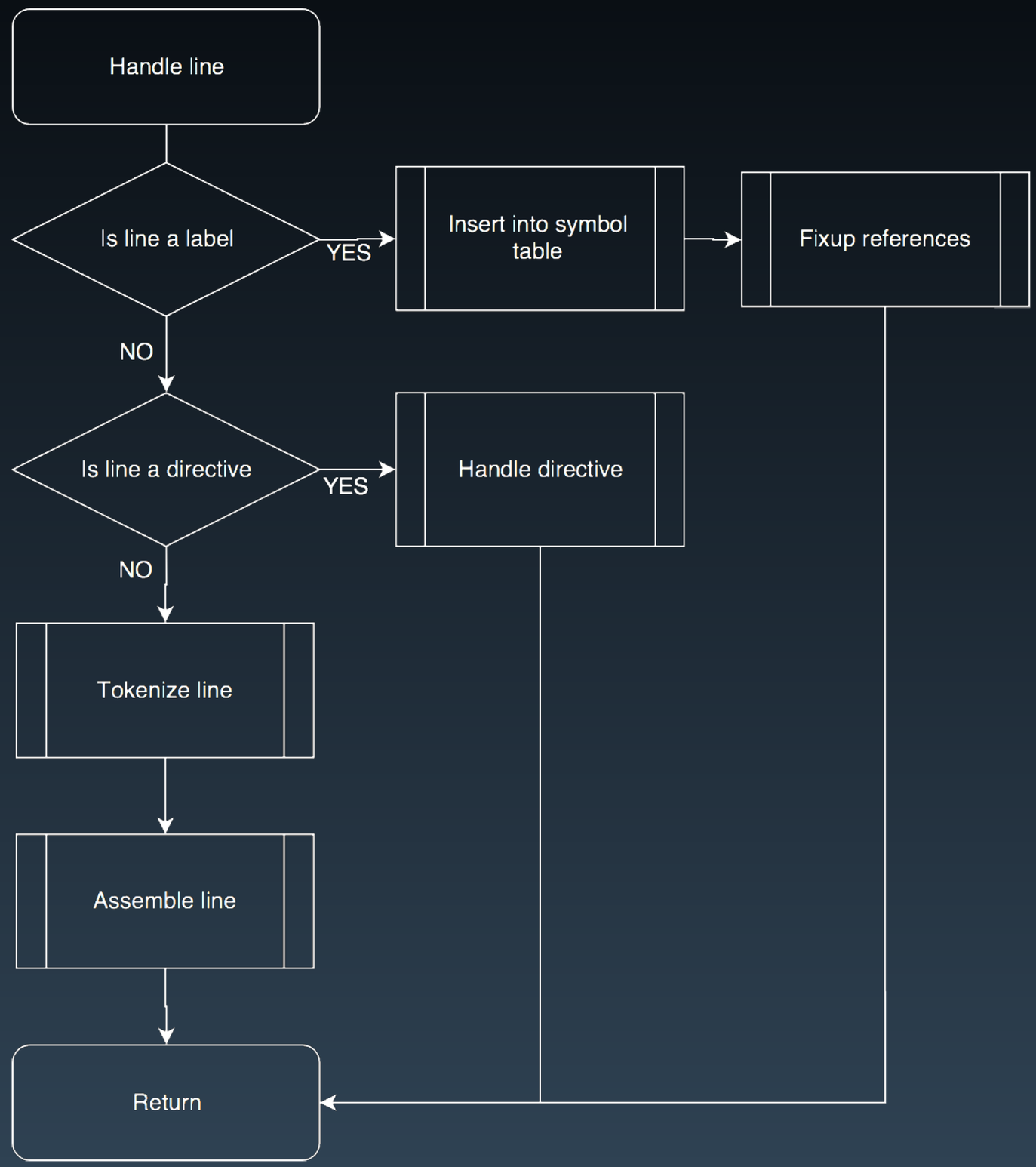
Symbol Table

- Label (char *) to a single uint32_t position of addr_ctr that we encountered it (for relative addressing)

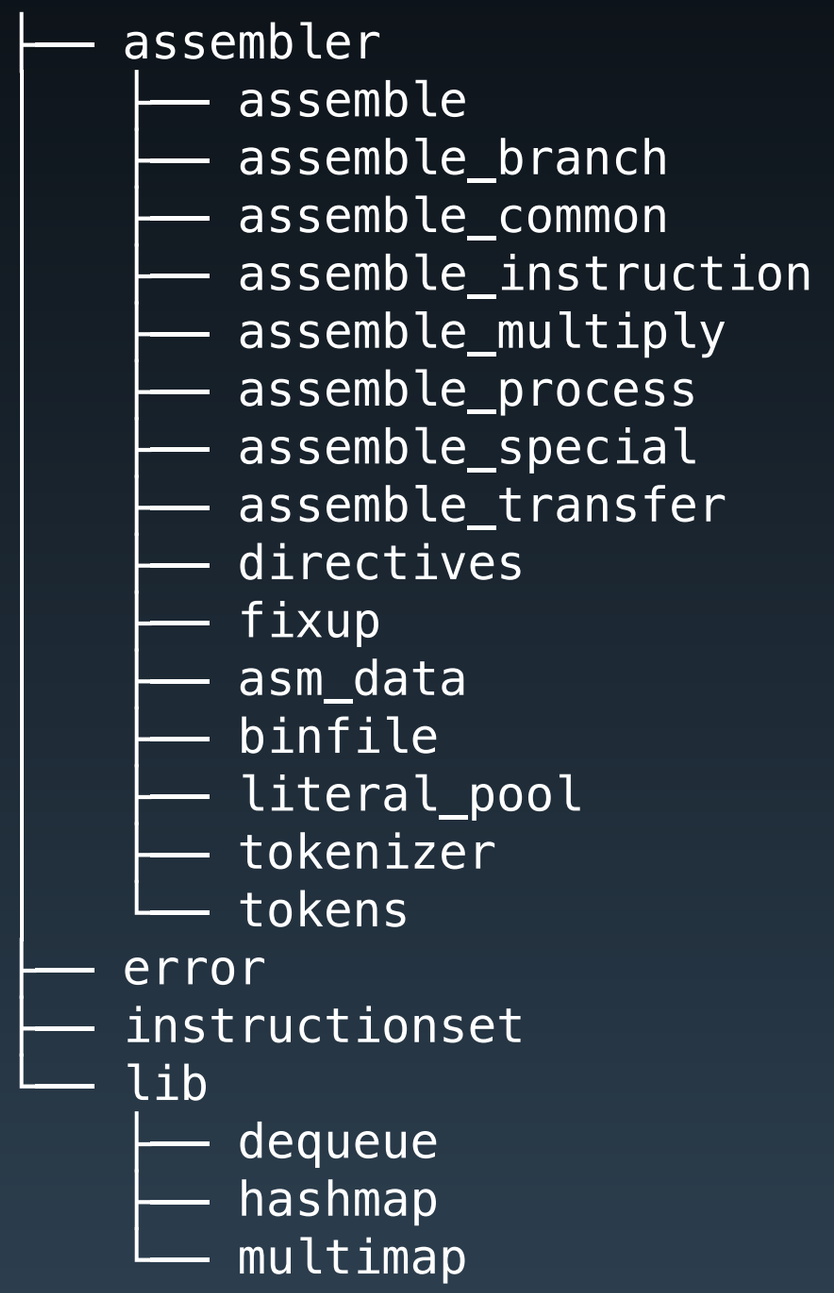
```
typedef struct {
    /** Address of the line to be fixed up */
    uint32_t addr;
    /** Address of resolved data */
    uint32_t resolved_addr;
} fixup_record;

/** @brief Callback for fixup functions */
typedef int (*fixup_cb)
    (asm_data *data, fixup_record *fixup);

/**
 * @brief struct to keep track of fix up requests
 */
typedef struct {
    /** function to call when data is resolved */
    fixup_cb callback;
    fixup_record record;
} fixup_request;
```


START
Initialise data structures
Lines remaining
NO
Handle EOF
Destroy data structures
END
YES
Read next line
Is line a comment or empty
YES
Handle line

Processes

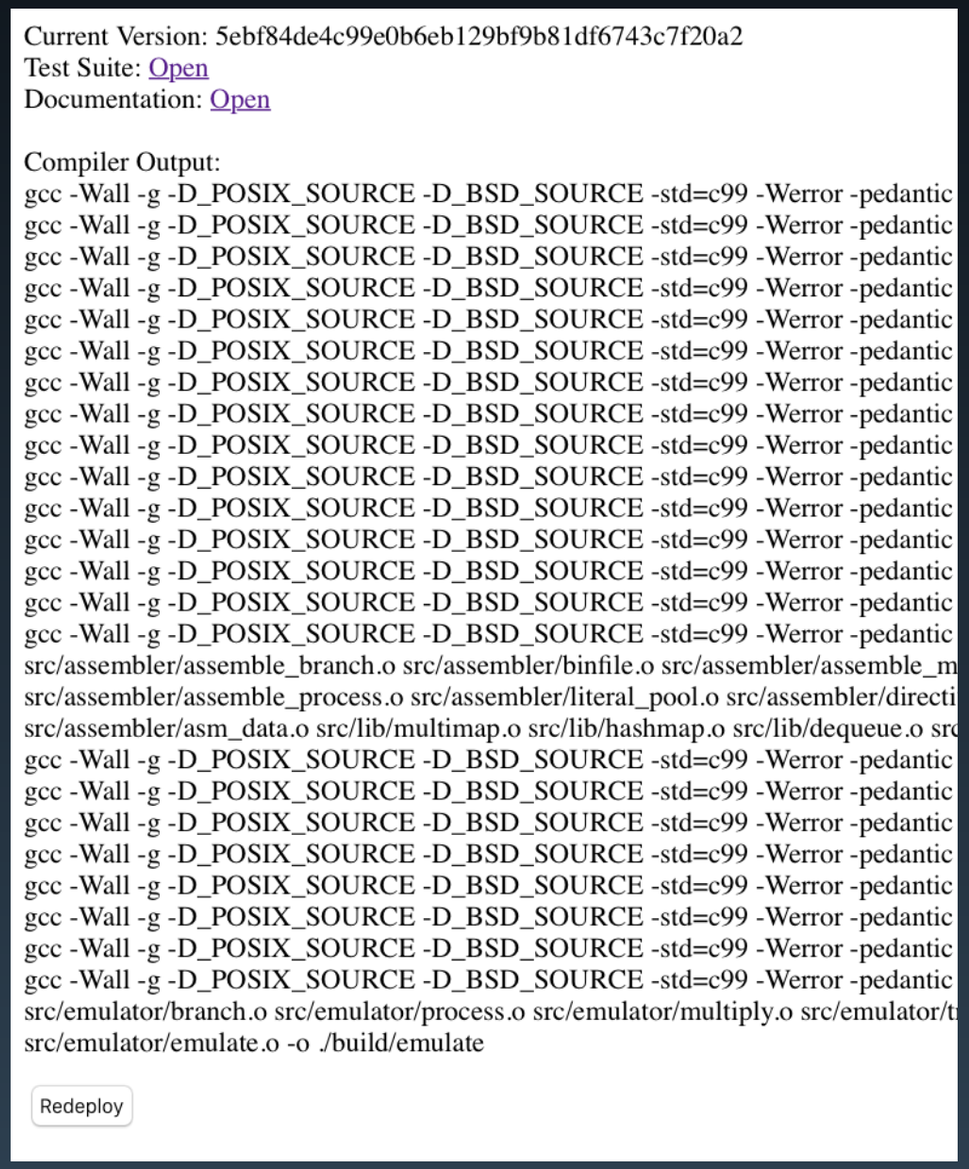

Handle line
Is line a label
YES
Insert into symbol table
Fixup references
NO
Is line a directive
YES
Handle directive
NO
Tokenize line
Assemble line
Return

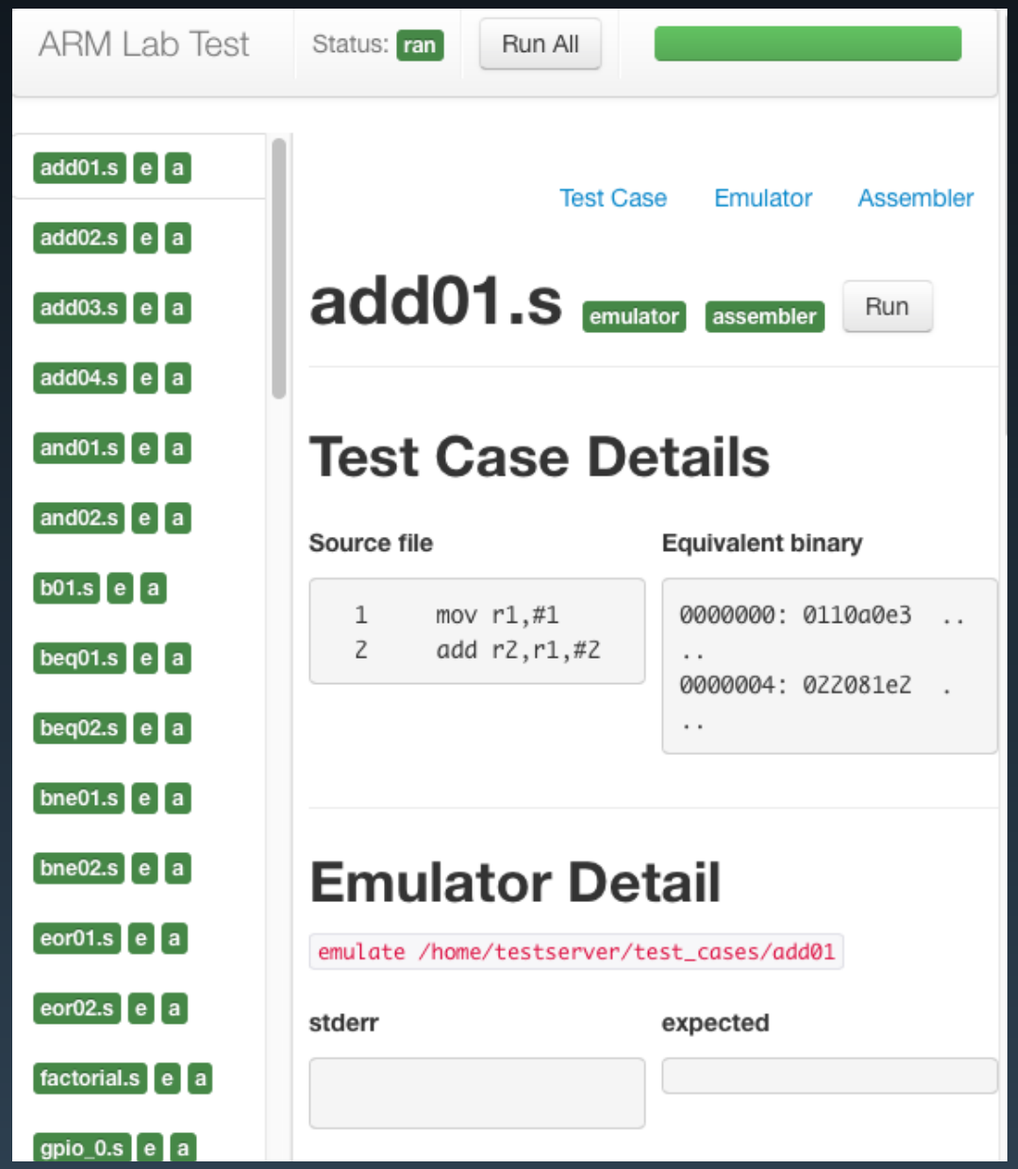
Processes


assembler
assemble
assemble_branch
assemble_common
assemble_instruction
assemble_multiply
assemble_process
assemble_special
assemble_transfer
directives
fixup
asm_data
binfile
literal_pool
tokenizer
tokens
error
instructionset
lib
dequeue
hashmap
multimap

Processes

Testing





Test Suite

```c
void tokenizer_test_one()
{
    /* Setup for test */
    token *tokens = malloc(32 * sizeof(token*));
    assert_true(tokens != NULL);

    /* Test */
    printf("mov r1,r2\n");
    int size = tokenize_line("mov r1,r2", tokens);

    /* Expected */
    token expected[4];
    expected[0].type          = OPERATION;
    expected[0].val.operation = MOV;
    expected[1].type          = OPERAND;
    expected[1].val.operand   = R1;
    expected[2].type          = OPERAND;
    expected[2].val.operand   = R2;
    expected[3].type          = TERMINATOR;

    /* Check */
    assert_int_equal(sizeof(expected) / sizeof(token), size);
    verify_tokens(tokens, expected, size);
    printf("\n");

    /* Free memory */
    free(tokens);
}
```

Tokenizer test case

```
------------ tests/tokenizer.c -------------
mov r1,r2
mov r6,#55
mov r9,#-42
mov [r1],=0xFF00000E
ldr r5,[r1], -r2,lsr #2
beq unicorn
ldr r3,[r1,#-0x4]
ldr r3,=unicorn
mov [bob],#unicorn
ldmia sp!, {r4-r12, lr}

10 run  0 failed

============== SEATEST v0.5 ==============

ALL TESTS PASSED
10 tests run
in 0 ms

==========================================
```

Sample output from running a test

Unit Testing

Debugging

```
pixel42% arm-linux-gnueabi-objdump -marm -b binary -D our_assembler > our_assembler_dump
pixel42% arm-linux-gnueabi-objdump -marm -b binary -D arm_assembler > arm_assembler_dump
pixel42% diff arm_assembler_dump our_assembler_dump
2c2
< arm_assembler:     file format binary
---
> our_assembler:     file format binary
10c10
<    8:e1510002 cmp  r1, r2
---
>    8:e1510000 cmp  r1, r2
23c23
<   3c:01540005 cmpeq    r4, r5
---
>   3c:01540000 cmpeq    r4, r5
53c53
<   b4:e8b001fc ldm  r0!, {r2, r3, r4, r5, r6, r7, r8}
---
>   b4:e8a001fc ldm  r0!, {r2, r3, r4, r5, r6, r7, r8}
```

Sample output

PiPong

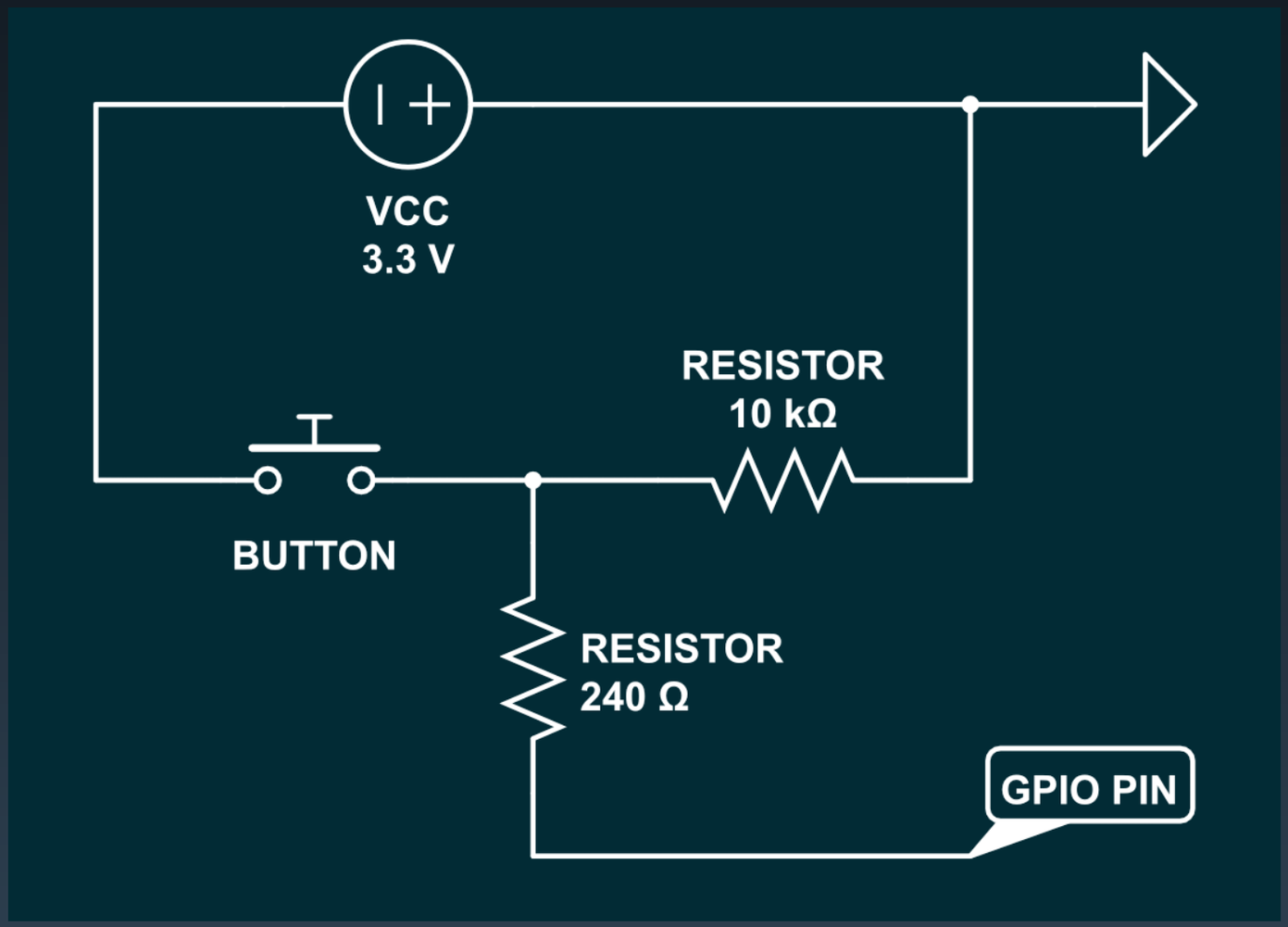
Game Design

The Game User Interface

```
game_loop:
    bl move_player_one
    bl move_player_two
    bl move_ball

    ldr r1, =player_one_score
    ldr r1, [r1]
    cmp r1, #11
    beq game_over

    ldr r1, =player_two_score
    ldr r1, [r1]
    cmp r1, #11
    beq game_over

    bl draw_scores
    bl draw_ball
    bl draw_player_one
    bl draw_player_two

    ldr r0, =game_background
    bl swap_buffers
b game_loop
```

The Game Loop in `main.s`

User Input

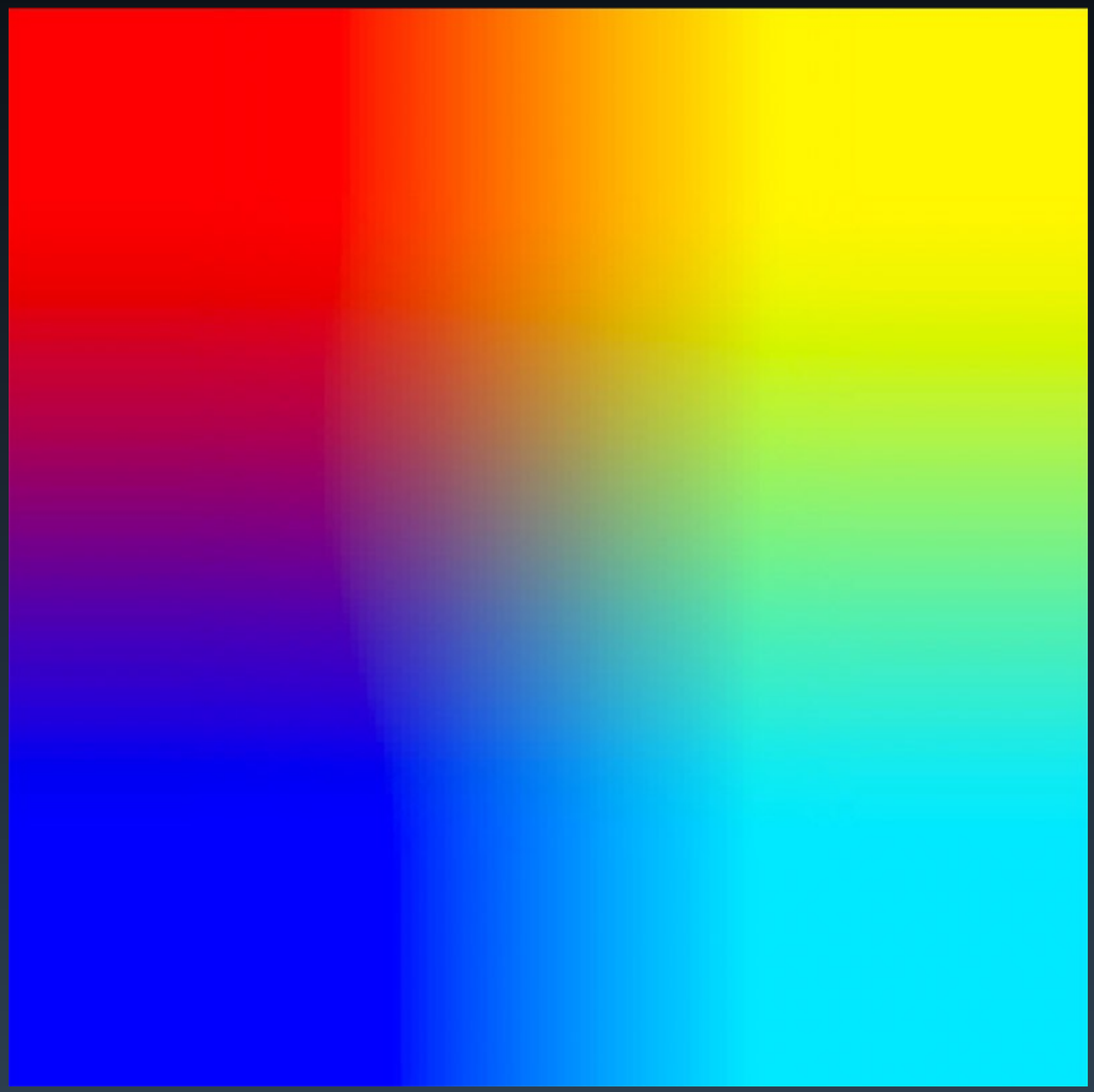


Controller button circuit diagram

Output

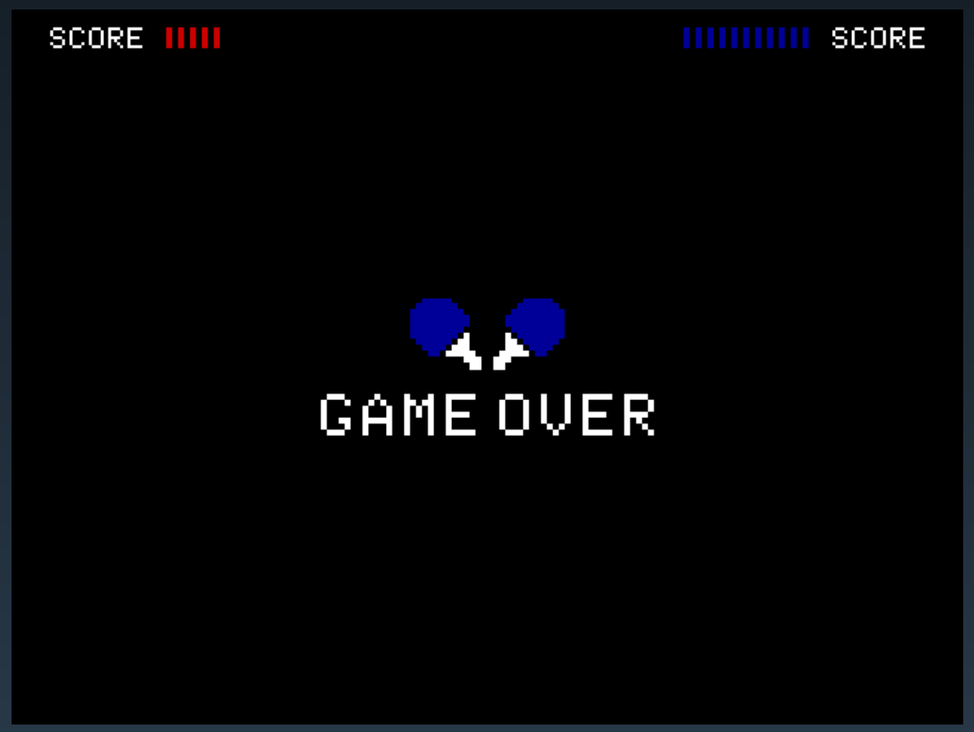
```
frame_buffer_info:
.4byte 640                      // +0x00 Physical width
.4byte 480                      // +0x04 Physical height
.4byte 640                      // +0x08 Virtual width
.4byte 480                      // +0x0c Virtual height
.4byte 0                        // +0x10 GPU pitch
.4byte 16                       // +0x14 GPU depth
.4byte 0                        // +0x18 X
.4byte 0                        // +0x1c Y
.4byte 0                        // +0x20 Buffer pointer
.4byte 0                        // +0x24 Buffer size
```

Frame Buffer description



“Rainbow Screen of Death”

Graphics



Start screen and Game Over screen

```
start_screen_text:
.4byte 0xffff0000
.4byte 0xffffffff
.4byte 0xffffffff
.4byte 0xffffffff
.4byte 0x0000ffff
.4byte 0x00000000
.4byte 0x00000000
.4byte 0xffff0000
.4byte 0xffffffff
.4byte 0xffffffff
.4byte 0x0000ffff
.4byte 0x00000000
.4byte 0x00000000
.4byte 0xffff0000
...
```

```
start_screen_logo:
.incbin start_screen_logo

start_screen_text:
.incbin start_screen_text

game_background:
.incbin background

game_over_player1:
.incbin game_over_player1

game_over_player2:
.incbin game_over_player2

score_text:
.incbin score_text
```

`.incbin` directives

Extending the Assembler

bl	set link register and branch
stmdb	store multiple decrement before
ldm/stm	load and store multiple data
ldmia	load multiple increment after
ldr r{n} =label	load runtime address of label
mvn/bic/cmn	additional data processing

```
swap_buffers1:
   ldm r0!, {r3-r12}
   stm r1, {r3-r12}
```

```
draw_player_one:
    stmdb sp!, {r4, lr}

    mov r0, #16
    ldr r1, =player_one_pos
    ldr r1, [r1]
    mov r2, #10
    mov r3, #80
    ldr r4, =0xC000C000
    bl draw_rect

    ldmia sp!, {r4, lr}
    mov pc, lr
```

Instructions

Extending the Assembler

.space fill with zeros
.4byte insert 32 bit integer
.ltorg save the literal pool
.incbin insert binary file

```
.space 60    // manually pad frame buffer
             // structure to 16 byte boundary

frame_buffer_info:
  .4byte 640  // +0x00 Physical width
  .4byte 480  // +0x04 Physical height
```

```
.ltorg  // dump the literal pool here to keep
        // image data labels in range

start_screen_logo:
.incbin start_screen_logo
```

Directives

Demo


A Group 12 Extension