Zachary Fraser

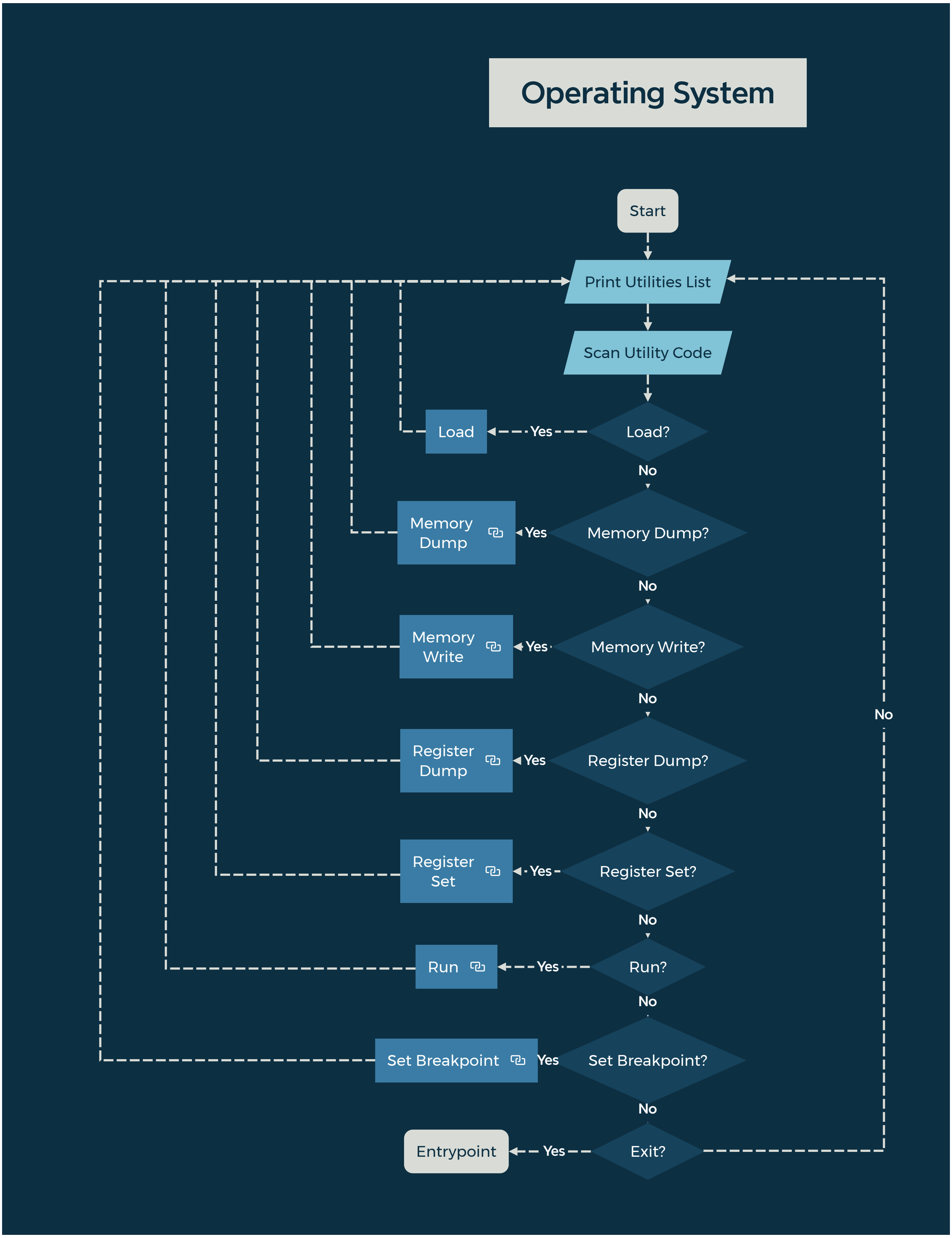
Lab 3 - Debugging Utilities

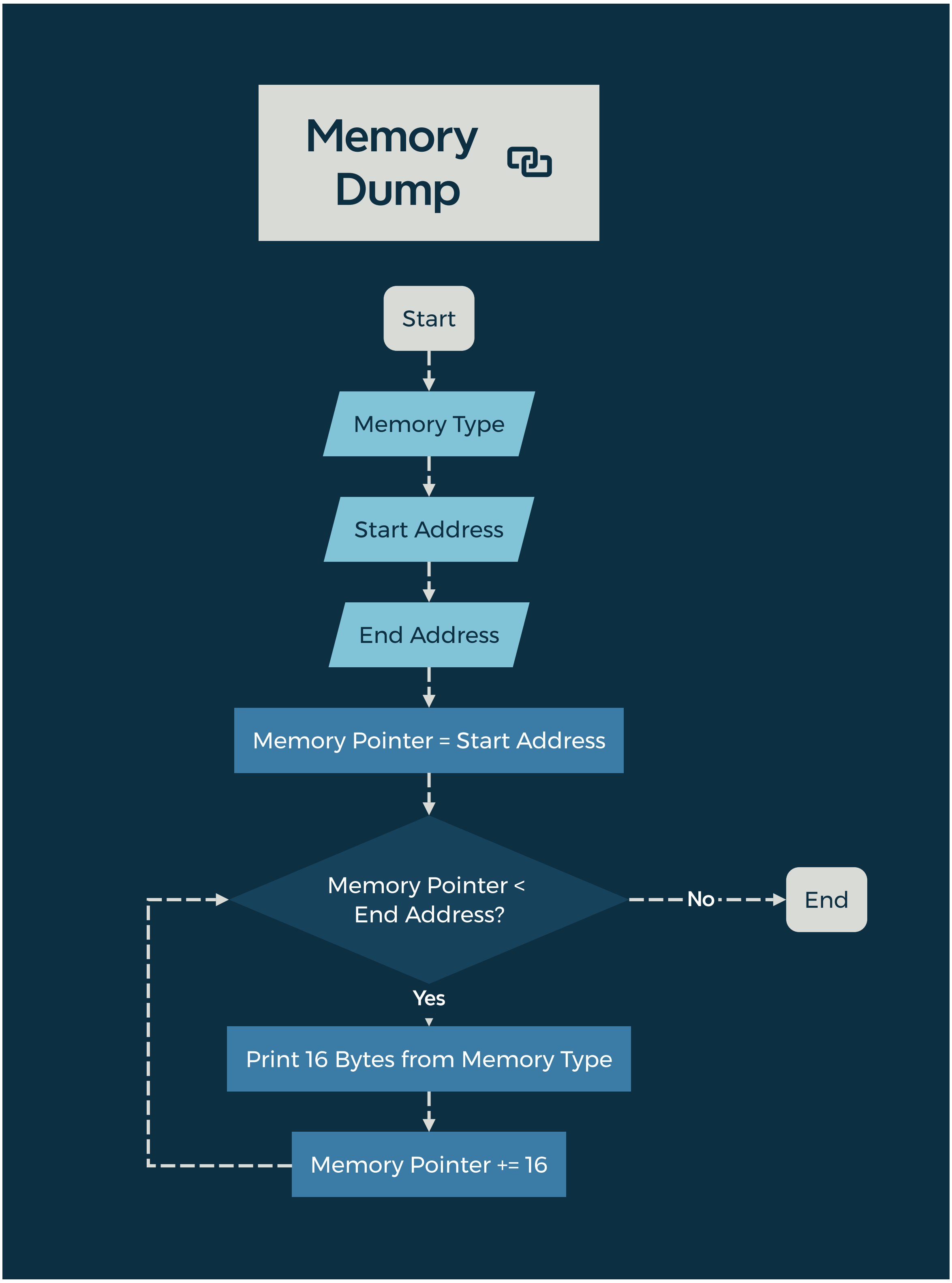
This lab implements several debugging utilities to the XM23P Emulator:

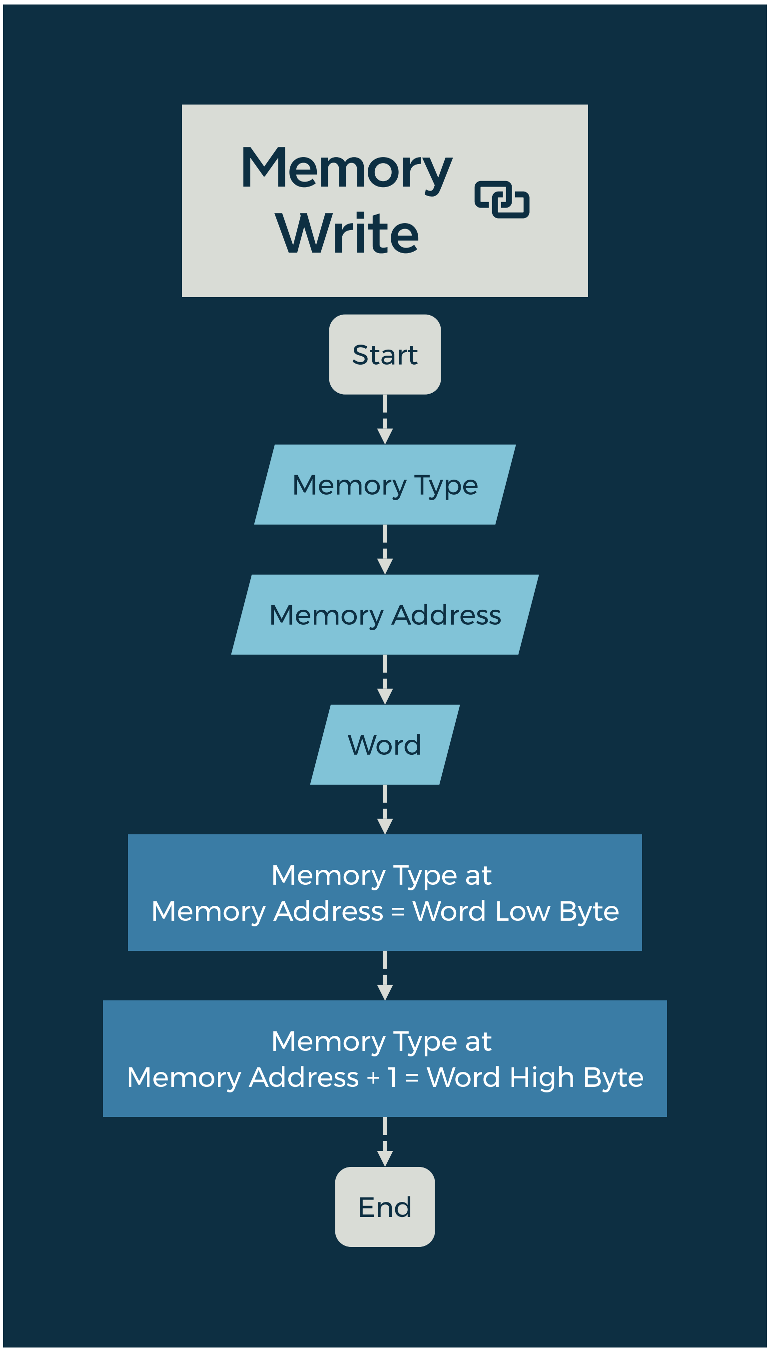
- Memory Read/Write
- Register Read/Write
- Breakpoint Insertion

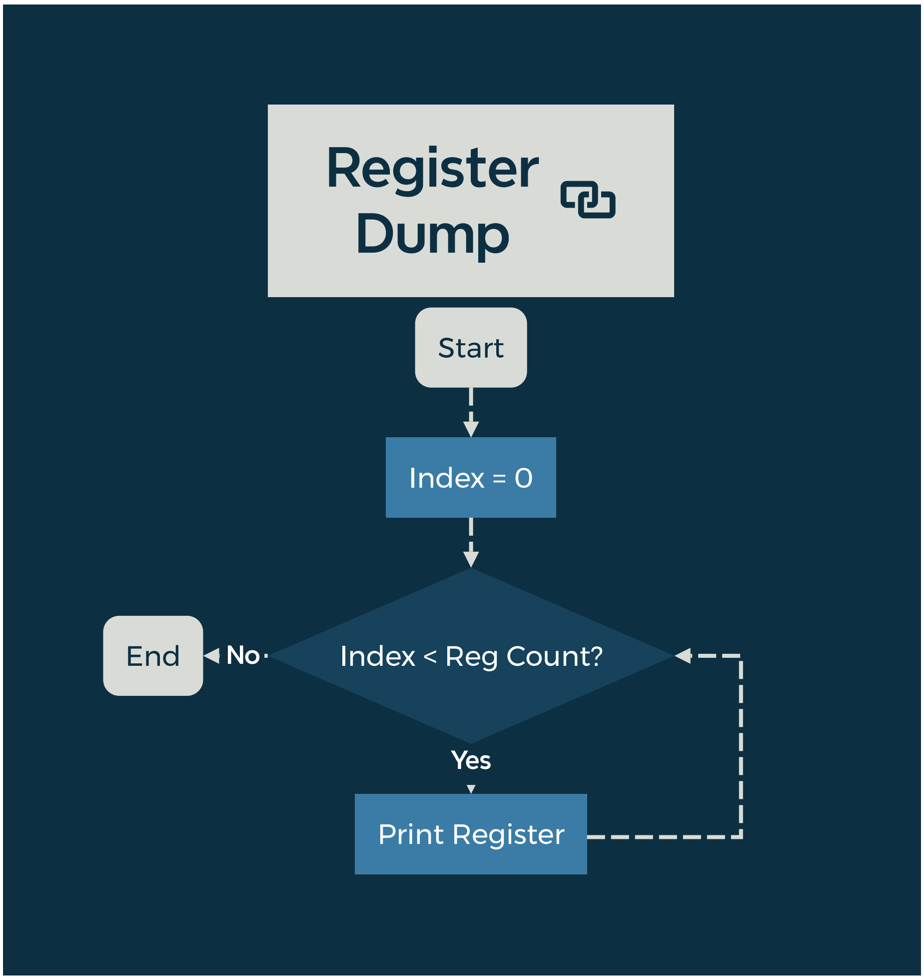
Design

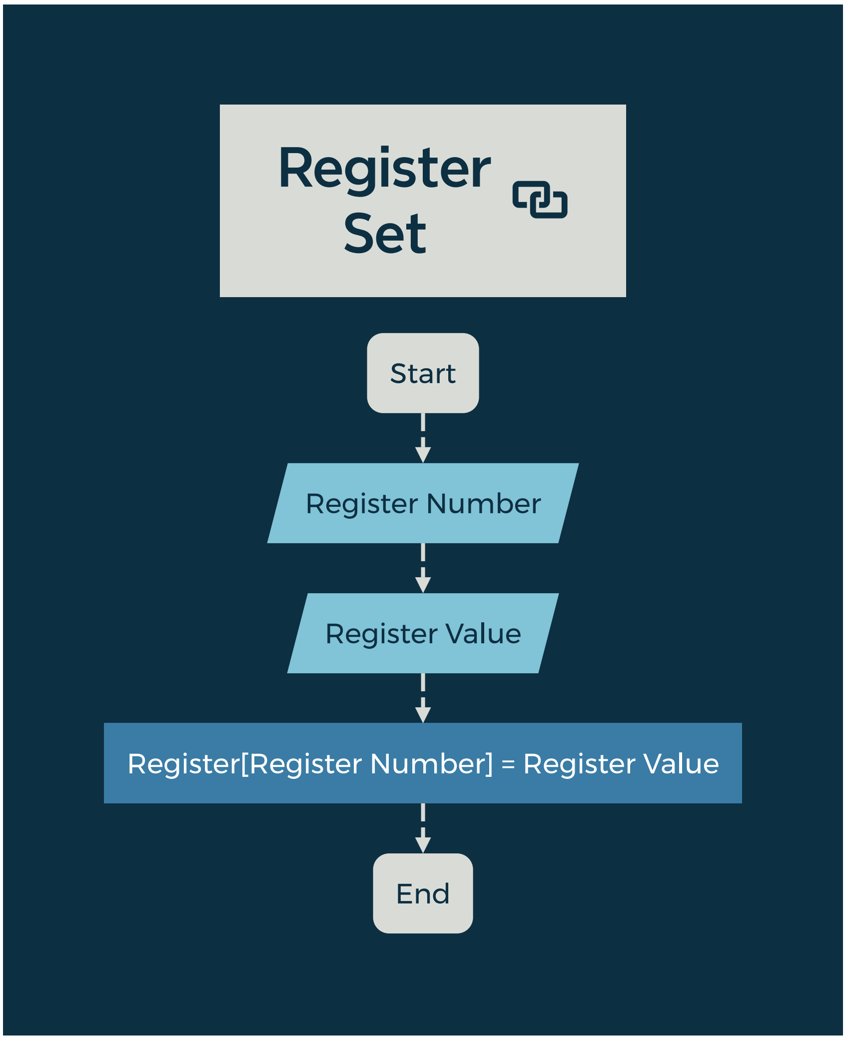
The design contains logic flowcharts detailing the implementation of a more robust "operating system", as well as the utilities.
A Data Dictionary for the current state of the emulator is included.

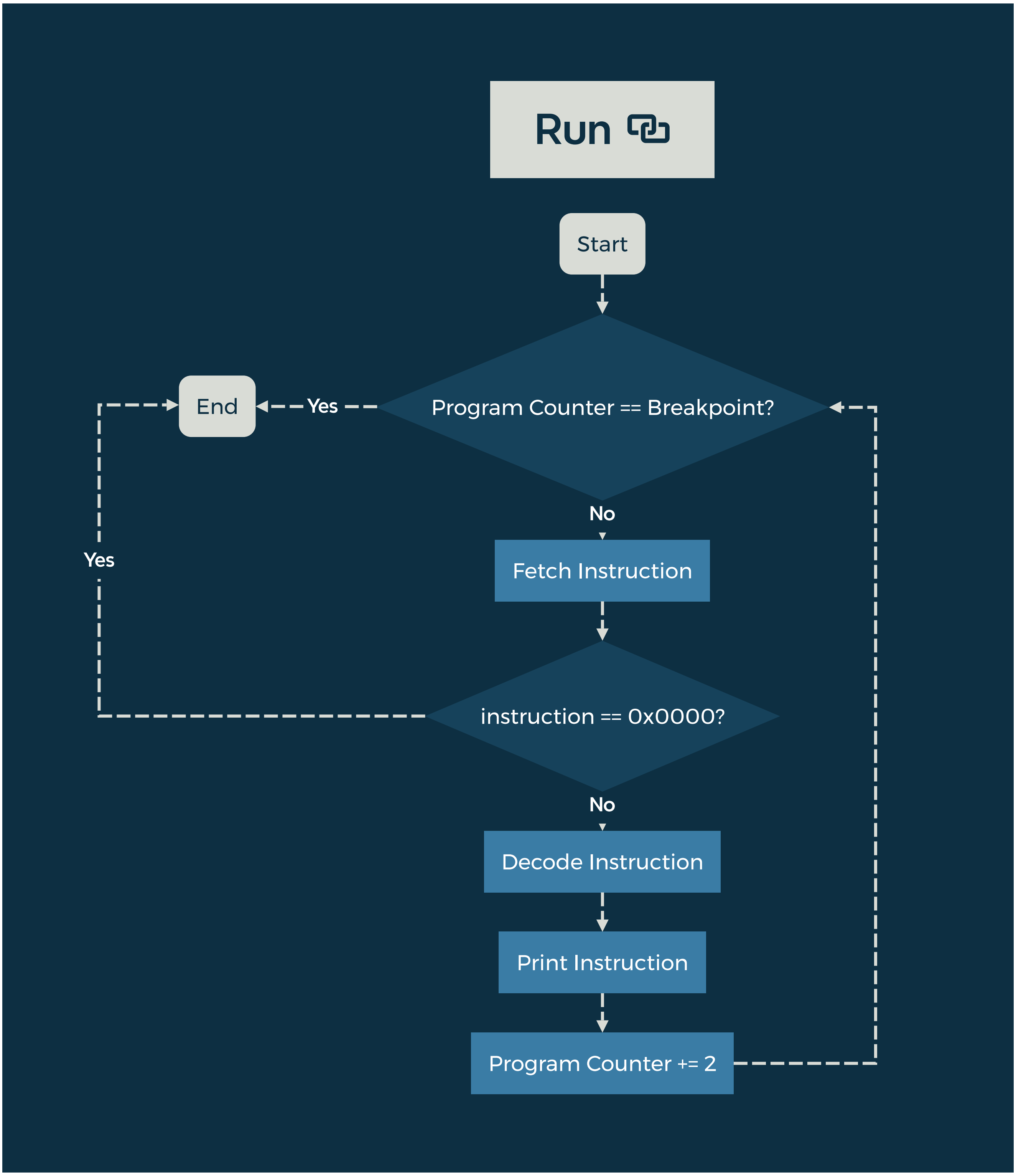

Operating System
Start
Print Utilities List
Scan Utility Code
Load?
Yes
Load
No
Memory Dump?
Yes
Memory Dump
No
Memory Write?
Yes
Memory Write
No
Register Dump?
Yes
Register Dump
No
Register Set?
Yes
Register Set
No
Run?
Yes
Run
No
Set Breakpoint?
Yes
Set Breakpoint
No
Exit?
Yes
Entrypoint
No

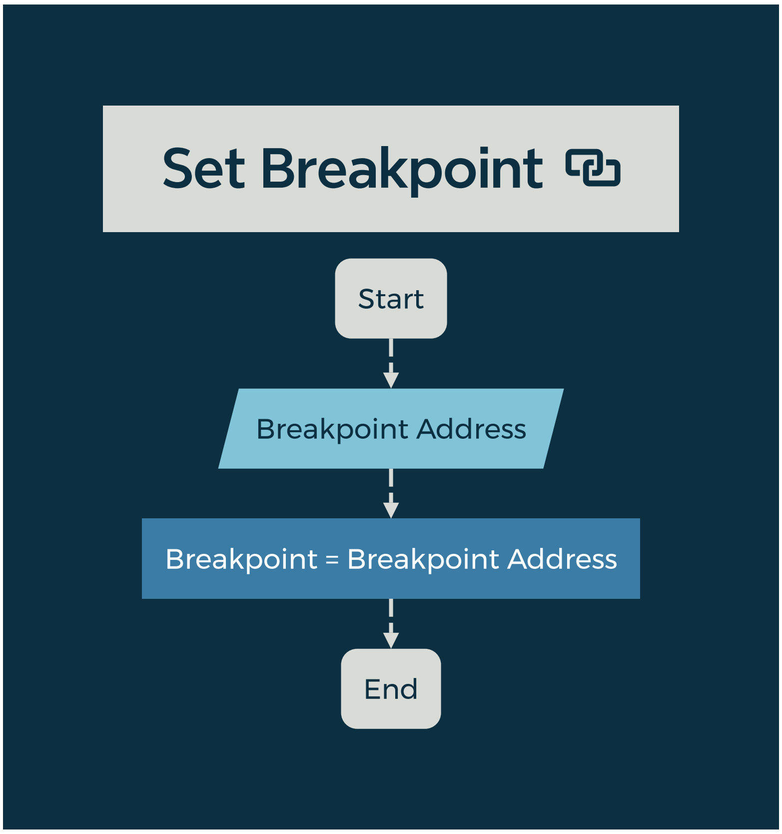

Memory Dump
Start
Memory Type
Start Address
End Address
Memory Pointer = Start Address
Memory Pointer < End Address?
No
End
Yes
Print 16 Bytes from Memory Type
Memory Pointer += 16


Memory
Write
Start
Memory Type
Memory Address
Word
Memory Type at
Memory Address = Word Low Byte
Memory Type at
Memory Address + 1 = Word High Byte
End


Register Dump
Start
Index = 0
Index < Reg Count?
No
End
Yes
Print Register


Register Set
Start
Register Number
Register Value
Register[Register Number] = Register Value
End


Run
Start
Program Counter == Breakpoint?
Yes
End
No
Fetch Instruction
instruction == 0x0000?
Yes
No
Decode Instruction
Print Instruction
Program Counter += 2


Set Breakpoint
Start
Breakpoint Address
Breakpoint = Breakpoint Address
End

Data Dictionary

```
PROGRAM         =   IMEM + DMEM + REGFILE + BREAKPOINT + START_ADDRESS + IR

IMEM            =   64*2^10{BYTE}64*2^10
DMEM            =   64*2^10{BYTE}64*2^10

REGFILE         =   8{WORD}8
BREAKPOINT      =   ADDRESS
START_ADDRESS   =   ADDRESS
IR              =   WORD

INSTRUCTION     =   CODE + 1{PARAMETER}4
CODE            =   [0-20] *Contiguous encoding of instructions*
PARAMETER       =   [RC|WB|SOURCE|DESTINATION|BYTE]

RC              =   BIT
WB              =   BIT
SOURCE          =   3{BIT}3
DESTINATION     =   3{BIT}3

ADDRESS         =   WORD
WORD            =   2{BYTE}2
BYTE            =   8{BIT}8
BIT             =   [0|1]

S_REC           =   'S' + REC_TYPE + LENGTH + ADDRESS + DATA
REC_TYPE        =   ['0'|'1'|'2'|'9']
LENGTH          =   BYTE_PAIR
ADDRESS         =   2{BYTE_PAIR}2
DATA            =   1{BYTE_PAIR}30
BYTE_PAIR       =   2{CHAR}2
CHAR            =   ['0'-'F']
```

Testing

The following tests were implemented:

- Test_10: Instruction Memory Dump
- Test_11: Data Memory Dump
- Test_12: Instruction Memory Write
- Test_13: Data Memory Write
- Test_14: Register Dump
- Test_15: Register Set
- Test_16: Breakpoint Set

Test_10: Instruction Memory Dump

Purpose
Test the printing of memory from instruction memory.
Configuration

1. Test10_Program_Debugging.xme was loaded into the emulator.
2. `m` was entered to start the Memory Dumping Utility.
3. `0` was entered to select Instruction Memory.
4. Start Address of `0x0100` was entered.
5. End Address of `0x0180` was entered.

Expected Results
The program should print the contents of memory between address `#0100` and `#0180`.
Results
The contents were correctly printed:

```
#0100   f0 65 79 7f d2 6f 73 76 01 40 d3 40 25 41 f7 41
#0110   01 42 d3 42 25 43 f7 43 01 44 d3 44 25 45 f7 45
#0120   01 46 d3 46 25 47 f7 47 01 48 d3 48 25 49 f7 49
#0130   01 4a d3 4a 25 4b f7 4b 01 4c 53 4c a5 4c 06 4d
#0140   47 4d 08 4d 49 4d 1a 4d 23 4d 0c 00 0b 20 0a 20
#0150   09 24 08 24 07 28 06 28 05 2c 04 2c 03 30 02 34
#0160   01 38 00 3c 81 4d 92 4d bf 4d df 4d 48 50 01 58
#0170   01 5c 82 80 81 c0 00 00 00 00 00 00 00 00 00 00
```

Pass/Fail
Pass.

Test_11: Data Memory Dump

Purpose

Test the printing of memory from data memory.

Configuration

1. Test10_Program_Debugging.xme was loaded into the emulator.
2. `m` was entered to start the Memory Dumping Utility.
3. `1` was entered to select Data Memory.
4. Start Address of `0x0800` was entered.
5. End Address of `0x0810` was entered.

Expected Results

The program should print the contents of memory between address `#0800` and `#0810`.

Results

The contents were correctly printed:

```
Memory Dump Utility
Select Memory Type:
0 - Program Memory | 1 - Data Memory
1
Enter Memory Start Address: 800
Enter Memory End Address: 810
#0800   fe ed be ef 00 00 00 00 00 00 00 00 00 00 00 00
```

Pass/Fail

Pass.

Test_12: Instruction Memory Write

Purpose

Test the writing of memory to instruction memory.

Configuration

1. Test10_Program_Debugging was loaded into the emulator.
2. `w` was entered to start the Memory Writing Utility.
3. `0` was entered to select Instruction Memory.
4. Start Address of `0x0100` was entered.
5. Data `0x1234` was entered.
6. `m` was entered to start the Memory Dumping Utility.
7. `0` was entered to select Instruction Memory.
8. Start Address of `0x0100` was entered.
9. End Address of `0x0180` was entered.

Expected Results

The program should write the data `0x1234` to memory address `#0100`.

Results

The data was successfully written to memory in little endian format:

```
Memory Dump Utility
Select Memory Type:
0 - Program Memory | 1 - Data Memory
0
Enter Memory Start Address: 100
Enter Memory End Address: 180
#0100   34 12 79 7f d2 6f 73 76 01 40 d3 40 25 41 f7 41
#0110   01 42 d3 42 25 43 f7 43 01 44 d3 44 25 45 f7 45
#0120   01 46 d3 46 25 47 f7 47 01 48 d3 48 25 49 f7 49
#0130   01 4a d3 4a 25 4b f7 4b 01 4c 53 4c a5 4c 06 4d
#0140   47 4d 08 4d 49 4d 1a 4d 23 4d 0c 00 0b 20 0a 20
#0150   09 24 08 24 07 28 06 28 05 2c 04 2c 03 30 02 34
#0160   01 38 00 3c 81 4d 92 4d bf 4d df 4d 48 50 01 58
#0170   01 5c 82 80 81 c0 00 00 00 00 00 00 00 00 00 00
```

Pass/Fail

Pass.

Test_13: Data Memory Write

Purpose

Test the writing of memory to data memory.

Configuration

1. Test10_Program_Debugging was loaded into the emulator.
2. `w` was entered to start the Memory Writing Utility.
3. `1` was entered to select Data Memory.
4. Start Address of `0x0200` was entered.
5. Data `0xABCD` was entered.
6. `m` was entered to start the Memory Dumping Utility.
7. `1` was entered to select Data Memory.
8. Start Address of `0x0200` was entered.
9. End Address of `0x0210` was entered.

Expected Results

The program should write the data `0xABCD` to memory address `#0200`.

Results

The data was successfully written to memory in little endian format:

```
Memory Dump Utility
Select Memory Type:
0 - Program Memory | 1 - Data Memory
1
Enter Memory Start Address: 200
Enter Memory End Address: 210
#0200   cd ab 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Pass/Fail

Pass.

Test_14: Register Dump

Purpose

Test the printing of register values.

Configuration

1. Test10_Program_Debugging was loaded into the emulator.
2. `g` was entered to run the program and load the Program Counter.
3. `r` was entered to start the Register Dumping Utility.

Expected Results

The program should print the values of all registers, the program counter (register 7) should hold a non-zero value.

Results

The register values were correctly printed:

```
Register Dump Utility
R0: 0000
R1: 0000
R2: 0000
R3: 0000
R4: 0000
R5: 0000
R6: 0000
R7: 0178
```

Pass/Fail

Pass.

Test_15: Register Set

Purpose

Test the setting of register values.

Configuration

1. Test10_Program_Debugging was loaded into the emulator.
2. `s` was entered to start the Register Setting Utility.
3. Register `1` was selected.
4. Data `0x5678` was entered.
5. `r` was entered to start the Register Dumping Utility.

Expected Results

The program should set the value of register `R1` to `0x5678`.

Results

The value of register `R1` was successfully set:

```
Register Dump Utility
R0: 0000
R1: 5678
R2: 0000
R3: 0000
R4: 0000
R5: 0000
R6: 0000
R7: 0178
```

Pass/Fail

Pass.

Test_16: Breakpoint Set

Purpose

Test the setting of breakpoints.

Configuration

1. Test10_Program_Debugging was loaded into the emulator.
2. `b` was entered to start the Breakpoint Setting Utility.
3. Breakpoint address `0x0110` was entered.
4. `g` was entered to Run the program.

Expected Results

The program should set a breakpoint at address `0x0110`, and execution should stop at this point.

Results

The breakpoint was successfully set, and execution stopped at `0x0110`:

```
Run Utility
Program Counter: 0100
Breakpoint: 0110
0100: 1234
0102: 7f79 - MOVH  RC: 0 WB: 0 Source: ef Destination: 01
0104: 6fd2 - MOVLZ RC: 0 WB: 0 Source: fa Destination: 02
0106: 7673 - MOVLS RC: 0 WB: 0 Source: ce Destination: 03
0108: 4001 - ADD   RC: 0 WB: 0 Source: 00 Destination: 01
010a: 40d3 - ADD   RC: 1 WB: 1 Source: 02 Destination: 03
010c: 4125 - ADDC  RC: 0 WB: 0 Source: 04 Destination: 05
010e: 41f7 - ADDC  RC: 1 WB: 1 Source: 06 Destination: 07
```

Pass/Fail

Pass