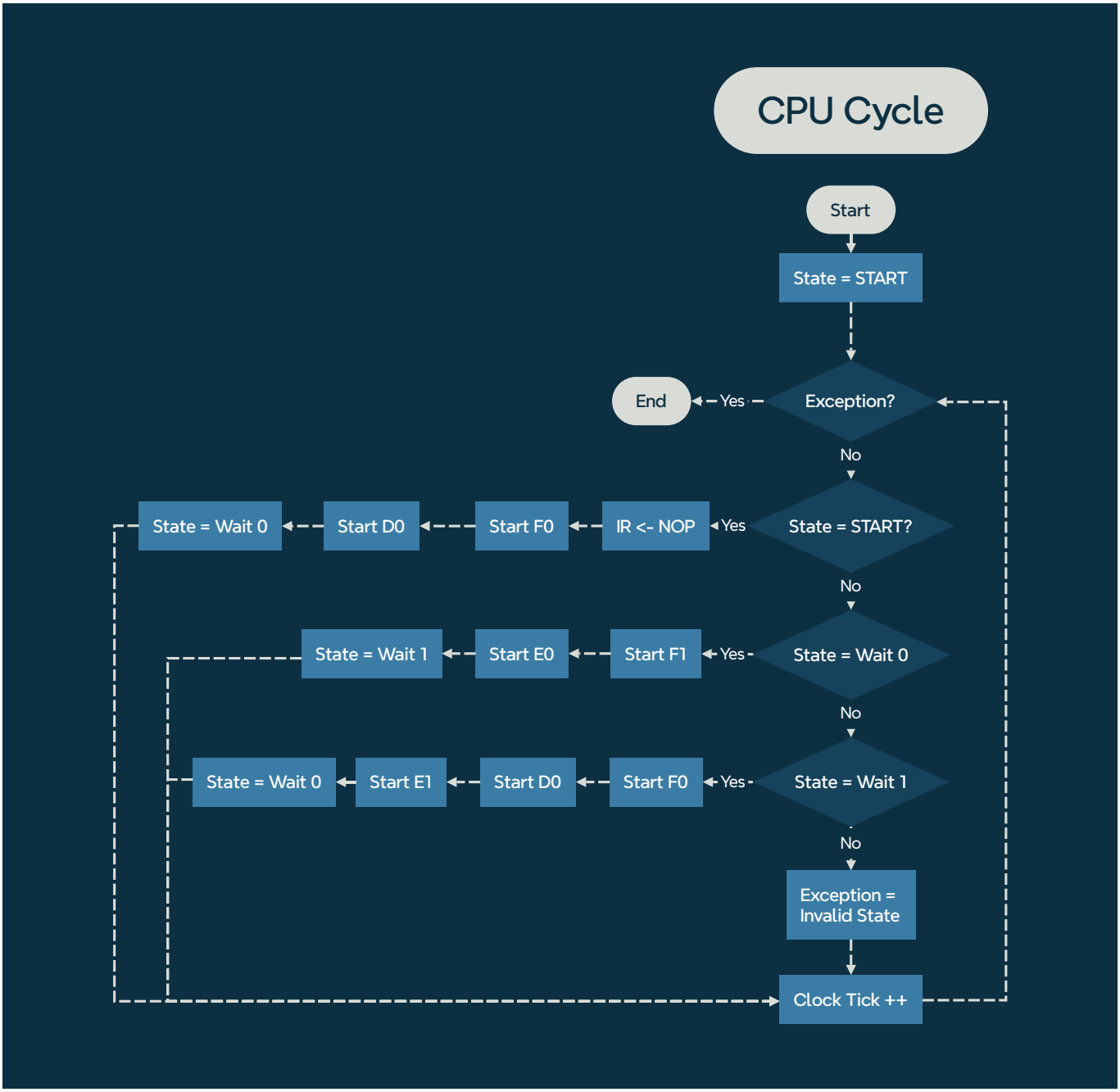
Zachary Fraser

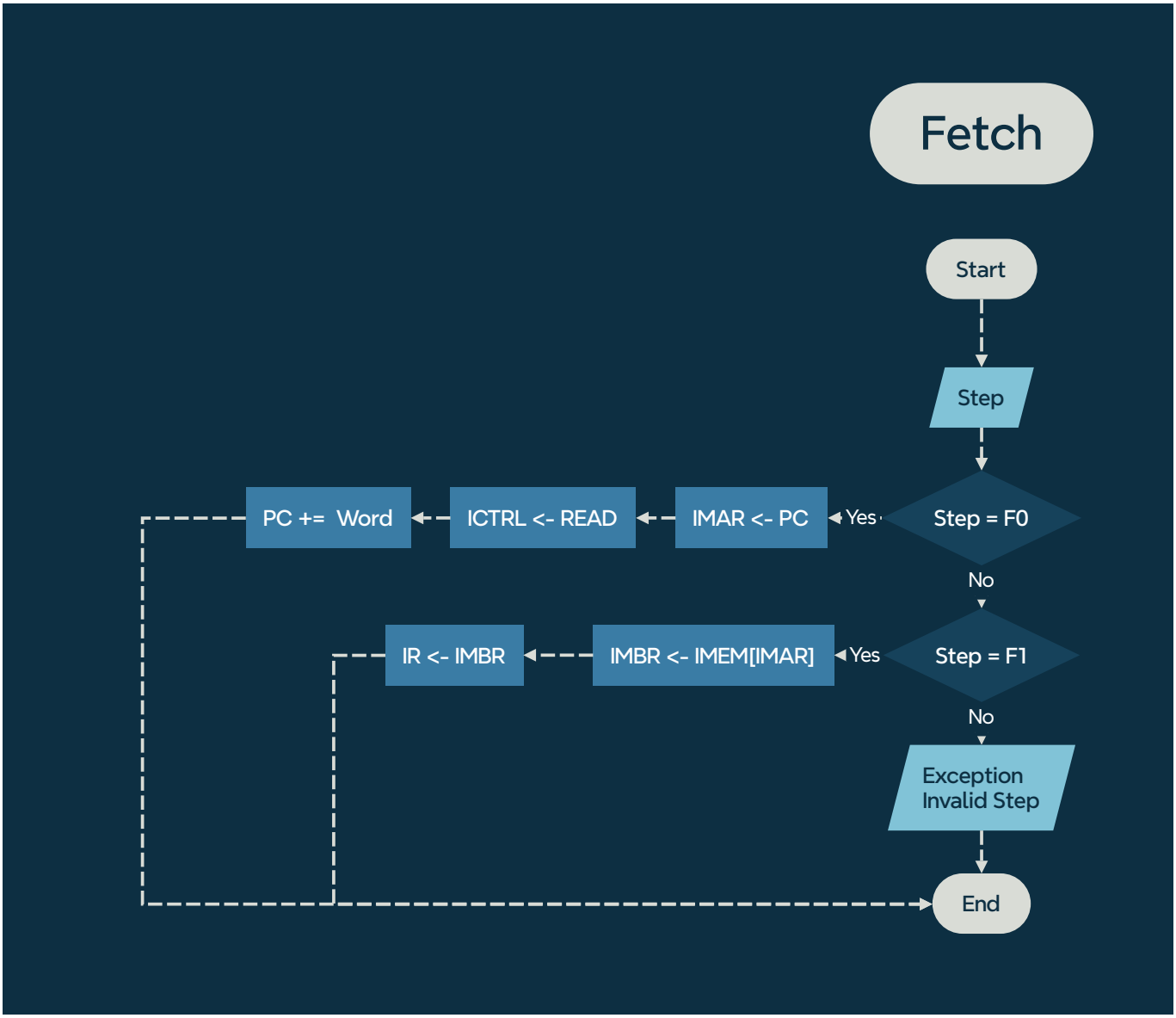
Assignment 3 - XM23P Data Pipeline

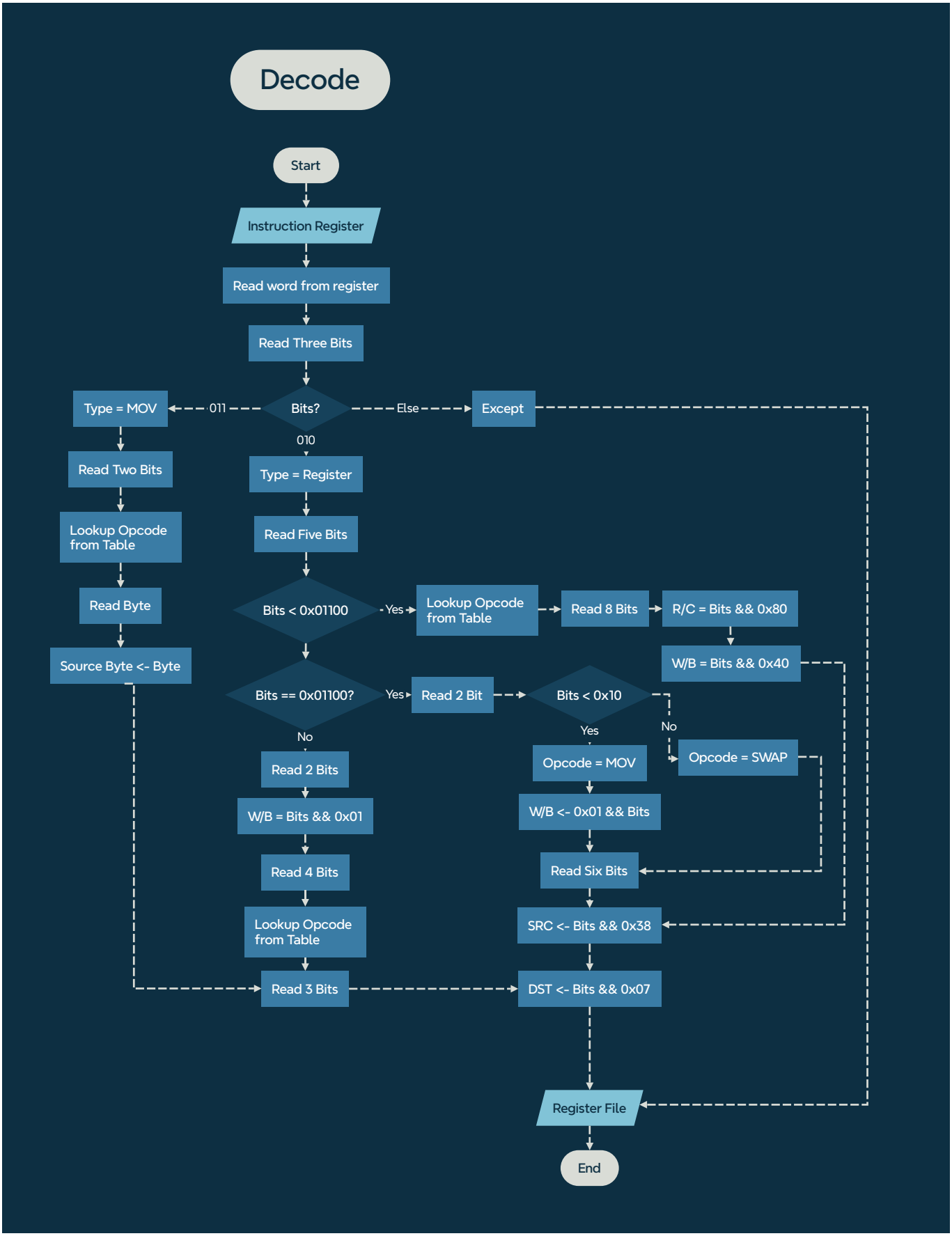
This assignment aims to implement the data memory access pipeline for the XM23P emulator. There are three stages: Fetch, Decode, and Execute. In this assignment, the Execute stage is extended to allow for data memory access in a similar manner to the instruction memory access seen in the fetch stage.

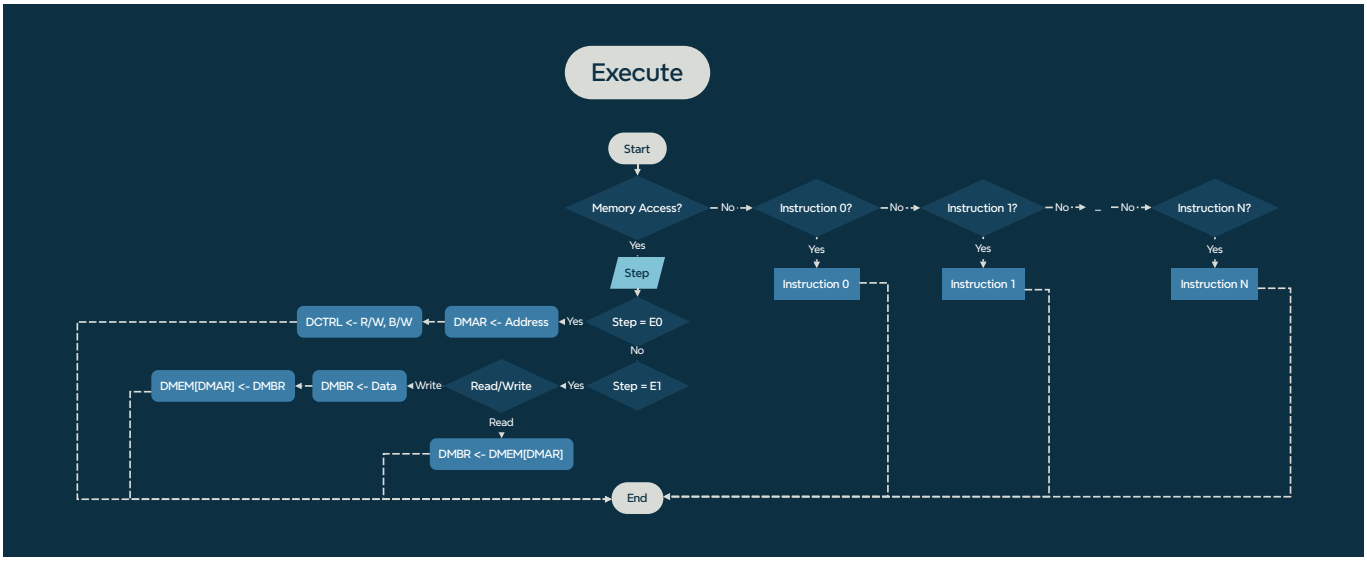
Design

The design contains logic flowcharts detailing the CPU Loop, and the Fetch, Decode, and Execute Stages. A Data dictionary describing the instructions and register file is also included.




Fetch
Start
Step
Step = F0
Yes
IMAR <- PC
ICTRL <- READ
PC += Word
No
Step = F1
Yes
IMBR <- IMEM[IMAR]
IR <- IMBR
No
Exception
Invalid Step
End


Decode
Start
Instruction Register
Read word from register
Read Three Bits
Bits?
011
Type = MOV
Else
Except
010
Read Two Bits
Type = Register
Lookup Opcode from Table
Read Five Bits
Read Byte
Bits < 0x01100
Yes
Lookup Opcode from Table
Read 8 Bits
R/C = Bits && 0x80
Source Byte <- Byte
W/B = Bits && 0x40
Bits == 0x01100?
Yes
Read 2 Bit
Bits < 0x10
No
Yes
No
Read 2 Bits
Opcode = MOV
Opcode = SWAP
W/B = Bits && 0x01
W/B <- 0x01 && Bits
Read 4 Bits
Read Six Bits
Lookup Opcode from Table
SRC <- Bits && 0x38
Read 3 Bits
DST <- Bits && 0x07
Register File
End


Execute
Start
Memory Access?
No
Instruction 0?
No
Instruction 1?
No
No
Instruction N?
Yes
Yes
Yes
Yes
Step
Instruction 0
Instruction 1
Instruction N
DCTRL <- R/W, B/W
DMAR <- Address
Yes
Step = E0
No
DMEM[DMAR] <- DMBR
DMBR <- Data
Write
Read/Write
Yes
Step = E1
Read
DMBR <- DMEM[DMAR]
End

Data Dictionary

```
IMEM            =   32*2^10{WORD}32*2^10
IMAR            =   ADDRESS
ICTRL           =   [READ|WRITE]
IMBR            =   WORD
IR              =   WORD

DMEM            =   64*2^10{BYTE}64*2^10
DMAR            =   ADDRESS
DCTRL           =   [READ|WRITE]
DMBR            =   WORD

REGFILE         =   3{GPR}3 + BP + LR + SP + PC
GPR             =   WORD *General Purpose Register*
BP              =   WORD *Base Pointer*
LR              =   WORD *Link Register*
SP              =   WORD *Stack Pointer*
PC              =   WORD *Program Counter*

PSW             =   PRV_PRI + 4{DC}4 + FLT + CUR_PRI + V + SLP + N + Z + C
PRV_PRI         =   3{BIT}3 *Previous Priority*
DC              =   BIT     *Don't Care*
FLT             =   BIT     *Fault*
CUR_PRI         =   3{BIT}3 *Current Priority*
V               =   BIT     *Arithmetic overflow*
SLP             =   BIT     *Sleep State*
N               =   BIT     *Negative Result*
Z               =   BIT     *Zero Result*
C               =   BIT     *Carry*

BREAKPOINT      =   ADDRESS
START_ADDRESS   =   ADDRESS

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
READ            =   0x0000
WRITE           =   0x0001

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