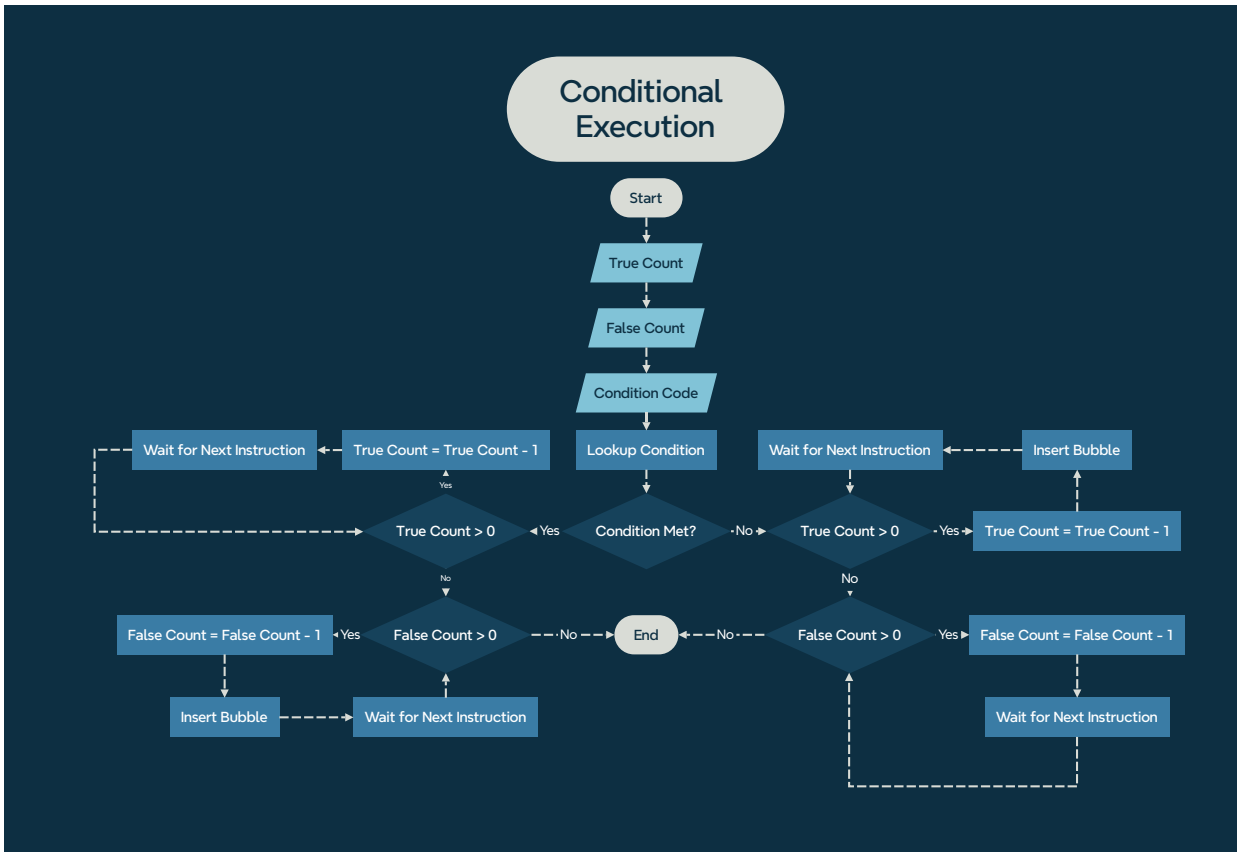
Zachary Fraser

Assignment 5 - Conditional Execution

This assignment aims to implement conditional execution of instructions other than branches using the CEX instruction. This will allow the XM23P to execute conditional code in a way that is less expensive than branching.

Design

The design contains logic flowcharts detailing the conditional execution. A Data dictionary describing the instructions, PSW, and register file is also included.



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{WORD}3 + BP + LR + SP + PC
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*

START_ADDRESS   =   ADDRESS

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

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

T_COUNT         =   [0-7]           *Number of instructions to execute if
condition is true*
F_COUNT         =   [0-7]           *Number of instructions to execute if
condition is false*
CONDITION_CODE  =   [#0000 - #1111] *Corresponds to execution condition*

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