ECED3403 – Assignment 3

Grace Yu

B00902046

July 2nd, 2024

1. Design

1.1. Problem Introduction

This assignment aims to further develop the XM23p emulator. The goal of assignment 3 is to design features that allow the emulator to perform four data memory access instructions. These four instructions are LD, LDR, ST, and STR. Accessing data memory involves a fifth stage, E1 (Execute 1), which will occur on even clock ticks. E1 will either write to or read memory, depending on what instruction called it.

1.2. Design Section

PSEUDOCODE:

```
A small amount of code used or referenced in the pseudocode was documented in
previous assignments or labs.

PIPELINE FUNCTION:
WHILE pc is not breakpoint AND instructionbit is not equal to 0
   IF clock/2 remainder is equal to 0
      IF DMAR is RD
         CALL execute1
      CALL fetch0
      CALL decode0
   ELSE
      CALL fetch1
      CALL execute0
   END IF
   IF increment mode is on AND clock/2 remainder is not equal to 0
      return
   END IF
   INCREMENT CLOCK
END FUNCTION

DECODE FUNCTION
   IF instruction opcode is LD or ST
      SAVE opcode
      SAVE PRPO, DEC, INC, W/B, SRC, and DST
   ELSE IF instruction opcode is LDR or STR
```

```
      SAVE opcode
      SAVE W/B, SRC, DST
   . . . other code from previous assignments and labs
   END IF
END FUNCTION

EXECUTE1 FUNCTION
   SWITCH(opcode)
      . . . other code from previous assignments and labs
      CASE LD
         CALL ld_execute
         BREAK
      CASE ST
         CALL st_execute
         BREAK
      CASE LDR
         CALL ldr_execute
         BREAK
      CASE STR
         CALL str_execute
         BREAK
   END SWITCH
END FUNCTION

ld_execute FUNCTION
   CALL reg_format
   IF pre-inc or pre-dec
      SRC = SRC + address_modifier
   END IF
   sign-extended offset = 0
   CALL ldx
   IF post-incr or post-dec
      SRC = SRC + address_modifier
   END IF
END FUNCTION

st_execute FUNCTION
   CALL reg_format
   IF pre-inc or pre-dec
      DST = DST + address_modifier
   END IF
   sign-extended offset = 0
   CALL stx
   IF post-incr or post-dec
      DST = DST + address_modifier
   END IF
END FUNCTION
```

```
ldx FUNCTION
   eff_address = SRC + sign-extended offset
   DMAR <- eff_address
   DCTRL <- RD
END FUNCTION

stx FUNCTION
   eff_address = DST + sign-extended offset
   DMAR <- eff_address
   DCTRL <- RD
END FUNCTION

reg_format FUNCTON
   IF inc is equal to 1
      SET address_modifier to 1
   ELSE IF dec is set to 1
      SET address_modifier to -1
   ELSE
      SET address_modifier to 0
   END IF
END FUNCTION

ldr_execute FUNCTION
   prepost-inc = 0
   prepost-inc = 0
   CALL ldx
END FUNCTION

str_execute FUNCTION
   prepost-inc = 0
   prepost-inc = 0
   call stx
END FUNCTION

execute1 FUNCTION
   SWITCH(opcode)
      CASE LD
         CALL ld_execute1
         BREAK
      CASE ST
         CALL st_execute1
         BREAK
      CASE LDR
         CALL ldr_execute1
         BREAK
      CASE STR
```

```
            CALL str_execute1
            BREAK
    END SWITCH
    DCTRL = DONE
END FUNCTION

ld_execute1 FUNCTION
    DMBR <- dmem[DMAR]
    DST <- DMBR
END FUNCTION

st_execute1 FUNCTION
    DMBR <- SRC
    dmem[DMAR] <- DMBR
END FUNCTION

ldr_execute1 FUNCTION
    DMBR <- dmem[DMAR]
    DST <- DMBR
END FUNCTIOn

str_execute1 FUNCTION
    DMBR <- SRC
    dmem[DMAR] <- SRC
END FUNCTION
```

1.3. Data Dictionary

```
prpo = [SET | CLEAR]
SET = 1
CLEAR = 0

dec = [SET | CLEAR]

inc = [SET | CLEAR]

SRC = [SET | CLEAR] + [SET | CLEAR] + [SET | CLEAR] * three bits *

DST = [SET | CLEAR] + [SET | CLEAR] + [SET | CLEAR] * three bits *

DMAR = [1 – 1<<15] * data memory address *

DMBR = [1 – 1<<15] * data memory buffer *

DCTRL = [READ | DONE]
```

```
READ = 1
DONE = 0
```