COMP2120 Assignment 5

YUAN Wenxuan (UID: 3036292740)

May 8, 2025

1. Consider a Serial Interface (e.g. Modem), containing a *Control & Status Register* and two *Buffer Registers*, Input and Output Buffer Register, residing in memory location `SCSR`, `SBRI` and `SBRO`,
   The `SCSR` has the following format:

```
Bit 0       =1 if Device Error
Bit 1       =1 if Device Ready
Bit 2       =0 if next operation is Write, 1, Read
Bit 3-5     =000 if speed = 4800 bps
            =001 if speed = 9600 bps
            =010 if speed = 19200 bps
            =011 if speed = 57600 bps
            =100 if speed = 115200 bps
Bit 6       =0 if odd parity, 1 if even parity
```

   Write an assembly program, using any instruction set (you may invent your own instructions) to output an array pf 10 characters by *Program I/O*, to the serial port, using a speed of 115200 bps and even parity. To simplify the problem, you may assume that the array of characters is stored in memory location `LINE`, with *one character in one word*. Only source program is needed.

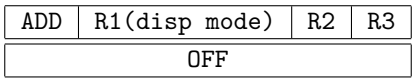
Solution:

```
        LD      SCSR, R1
        AND     R1, #0x1, R2    # check bit 0 device error
        BNZ     END             # if error, goto END
CHECK:  LD      SCSR, R1
        AND     R1, #0x2, R2    # check bit 1 device ready
        BZ      CHECK           # if not ready, loop and wait
        ST      #0x4a, SCSR     # bit pattern 100 1010
                                # 115200bps, even parity, write, ready
        SUB     R1, R1, R1      # R1 = 0
RUN:    LD      LINE(R1), R2    # load char from LINE[R1]
        ST      R2, SBRO        # write to output buffer register
        ADD     R1, #0x4, R1    # R1 += 4 (assume 1 word = 4 bytes)
        SUB     R1, #0x28, R2   # check if 10 chars (40 bytes) are written
        BNZ     RUN             # if R1 != 10, goto RUN (continue loop)
END:    HLT                     # write completed or escape due to error
```

2. Given the data path of a CPU as in Assignment 4 with the modification that the `MBR` provides data to both `S1-Bus` and `S2-Bus`. Consider another instruction set, which allows memory operands, and the addressing mode information is stored in the same byte as the register operand. Describe the data transfer/transformation for the following 2-word instruction:

```
ADD     OFF(R1), R2, R3
```

which will get the first operand from memory whose address is given by `OFF+R1` (displacement addressing mode), add it to `R2` and put the result in `R3`. `OFF` is stored in the word following the instruction:



Solution:

```
Fetch:
MAR <- PC
IR <- mem[MAR]
PC <- PC + 4

Execute:
MAR <- PC           # read OFF
MBR <- mem[MAR]
PC <- PC + 4
RFOUT1 <- R1
A <- RFOUT1
B <- MBR
C <- A + B          # EA = OFF + R1
MAR <- C
MBR <- mem[MAR]
A <- MBR            # operand 1: mem[OFF + R1]
RFOUT2 <- R2
B <- RFOUT2         # operand 2: R2
C <- A + B
RFIN <- C           # save result to R3
R3 <- RFIN
```