Department of Computer Science

Assignment #2

Shaheer Ziya
Roll: *3035946760*
Class: *2B*
Session: *2021-22*
Email: *shaheer@connect.hku.hk*

Course: *COMP 2120* – Teacher: *Dr. K P Chan, Wang Tianqi & and Xie Zerong*
Submission date: *Mar 14, 2022*

Part I: Example Program

```
        LD      P0,R4                   0000:  0600ff04  0000003c
        LD      P1,R1                   0008:  0600ff01  00000040
        MOV     R1,R2                   0010:  05010002
        LD      P2,R3                   0014:  0600ff03  00000044
L:      ADD     R4,R1,R4                001C:  00040104
        ADD     R1,R2,R1                0020:  00010201
        SUB     R3,R1,R5                0024:  01030105
        BNZ     L                       0028:  0802ff00  0000001c
        ST      R4,P                    0030:  0704ff00  00000048
        HLT                             0038:  09000000
P0:     .WORD   0                       003C:  00000000
P1:     .WORD   1                       0040:  00000001
P2:     .WORD   A                       0044:  0000000a
P:      .WORD                           0048:  00000000
```

What does the program above do?

Solution

The program calculates the sum $\Sigma_{r=1}^{9} r$, which in this case was $1+2+\cdots+8+9=45$ [1].

```python
# prog.py
# High-Level Code for prog (Assembly)
# Created by Shaheer Ziya on 24-02-2022 UTC+08 16:03

# Load values from memory to registers
R4 = 0
R1 = 1
R2 = R1
R3 = 10

# Initialize R5 w/out declaring it
R5 = None
# GoTo (loop) with Branch Condition R5 != 0
while (R5 != 0):
    R4 += R1          # Add value of Register 1 to Register 4
    R1 += 1           # Increment Register 1
    R5 = 10 - R1      # Store how many iterations left
# Halt Program

# Store value of Register 4 to Memory
print(f"The program calculates the sum 1 + 2 + ... + {R3 - 2} + {R3 - 1} = {R4}")
# Prints 'The program calculates the sum 1 + 2 + ... + 8 + 9 = 45'
```

[1] The code for the SUB and ST functions can be found in the Appendix

Part II: Hand Assemble

Solution

```
        LD      P0,R4               0000:   0600FF04  0000003C
        LD      P1,R1               0008:   0600FF01  00000040
        LD      P2,R2               0010:   0600FF02  00000044
        LD      P3,R3               0018:   0600FF03  00000048
L:      ADD     R4,R2,R4            0020:   00040204
        SUB     R3,R1,R3            0024:   01030103
        BNZ     L                   0028:   0802FF00  00000020
        ST      R4,P                0030:   0704FF00  0000004C
        HLT                         0038:   09000000
P0:     .WORD   0                   003C:   00000000
P1:     .WORD   1                   0040:   00000001
P2:     .WORD   5                   0044:   00000005
P3:     .WORD   4                   0048:   00000004
P:      .WORD                       004C:   00000000
```

The final result of the program is `0x14` or 20_{10}. The program multiplies the values in R2 & R3, then stores the result in R4 as abstracted in the python3 code below.

```python
# prog2.py
# High-Level code for Prog2 (Assembly)
# Created by Shaheer Ziya on UTC+08 17:12

# Load values from memory to registers
R4 = 0
R1 = 1
R2 = 5
R3 = 4

# GoTo (loop) with Branch Condition R3 != 0
while (R3 != 0):
    R4 += 5          # Add R2 to R4 (initially 0) 4 times (the initial value of R3)
    R3 -= 1          # Decrements R3; R3 is the number of iterations left
# Halt Program

# Store value of Register 4 to Memory
print(f"{R2} X 4 = {R4}")
# Prints '5 X 4 = 20'

# The program calculates R2 X R3 & writes the result to memory
```

Appendix: Code for SUB & ST

```
def set_SUB():
  Signal["calc_addr"] = 0
  Signal["branch"] = 0
  Signal["read_RF_port_1"] = 1
  Signal["read_RF_port_2"] = 1
  Signal["write_RF"] = 1
  Signal["src_of_S1"] = "RFOUT1"
  Signal["dst_of_S1"] = "A"
  Signal["src_of_S2"] = "RFOUT2"
  Signal["dst_of_S2"] = "B"
  Signal["src_of_D"] = "C"
  Signal["dst_of_D"] = "RFIN"
  Signal["doalu"] = 1
  Signal["ALU_func"] = "OP_SUB"
  Signal["move_via_S1"] = 1
  Signal["move_via_S2"] = 1
  Signal["move_via_D"] = 1
  Signal["read_memory"] = 0
  Signal["write_memory"] = 0
  Signal["dohalt"] = 0
```

The Subtraction (set-SUB) function reads inputs from RFOUT1 & RFOUT2, moves them to the buffers A & B through the busses S1 & S2 respectively. The ALU performs OP-SUB on the two operands, moves the result to the buffer C which is moved to RFIN through the D-Bus and stored in the destination register.

```
def set_ST():
  Signal["calc_addr"] = 1
  Signal["branch"] = 0
  Signal["read_RF_port_1"] = 1
  Signal["read_RF_port_2"] = 0
  Signal["write_RF"] = 0
  Signal["src_of_S1"] = "RFOUT1"
  Signal["dst_of_S1"] = "A"
  Signal["src_of_S2"] = ""
  Signal["dst_of_S2"] = ""
  Signal["src_of_D"] = "C"
  Signal["dst_of_D"] = "MBR"
  Signal["doalu"] = 1
  Signal["ALU_func"] = "OP_COPY"
  Signal["move_via_S1"] = 1
  Signal["move_via_S2"] = 0
  Signal["move_via_D"] = 1
  Signal["read_memory"] = 0
  Signal["write_memory"] = 1
  Signal["dohalt"] = 0
```

The Store (set-ST) function reads the data from the Register Files. It is output from RFOUT1 (which could also have been accomplished through RFOUT2) and moved to the buffer A through the S1-BUS. The ALU copies the data over to C from where the data is transferred to the MBR through the D-Bus and written to memory. (Of course the address for the storage of data in memory is also calculated in this step)