CPE 233: Software assignment 7

Prof. Bridget Benson

Luis Gomez, Marina Dushenko

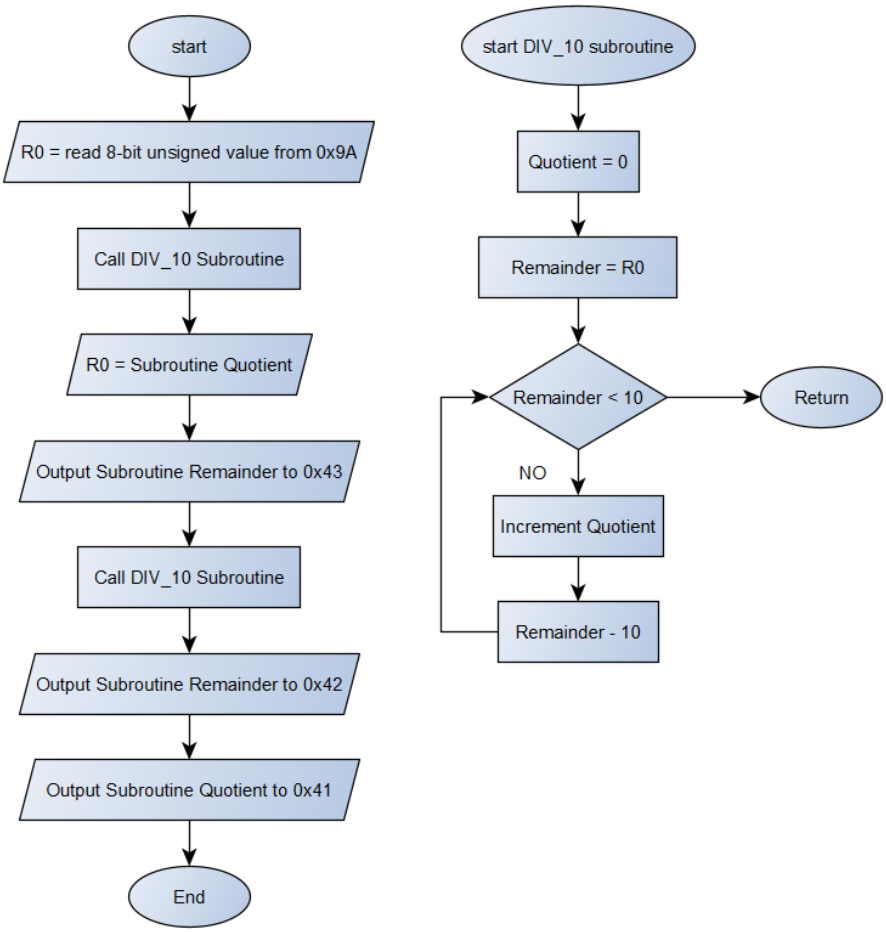
Behavior

In this assignment, we wrote two Assembly programs and subroutines using the RAT simulator.

Program 1: A Rat Assembly 8-bit BCD converter. The program reads an 8-bit value from port 0x9A and utilizes the DIV_10 subroutine twice, finally outputting the BCD equivalent of the 8-bit values via ports 0x41, 0x42, and 0x43.

Flowchart

Program 1



Verification

Program 1 Verification

		Registers			Output		
	In_Port 0x9A	**R0 (Input)**	**R1 (Quotient)**	**R2 (Remainder)**	**0x41**	**0x42**	**0x43**
start	0xFF	-	-	-	-	-	-
1st Call	-	0xFF	0x19	0x05	-	-	-
2nd Call	-	0x19	0x02	0x05	-	-	0x05
end	-	0x19	0x02	0x05	**0x02**	**0x05**	**0x05**
start	0x68	-	-	-	-	-	-
1st Call	-	0x68	0x0A	0x04	-	-	-
2nd Call	-	0x0A	0x01	0x00	-	-	0x04
end	-	0x0A	0x01	0x00	**0x01**	**0x00**	**0x04**
start	0x09	-	-	-	-	-	-
1st Call	-	0x09	0x00	0x09	-	-	-
2nd Call	-	0x00	0x00	0x00	-	-	0x09
end	-	0x00	0x00	0x00	**0x00**	**0x00**	**0x09**

Source Code

PROGRAM 1: 8b BCD converter

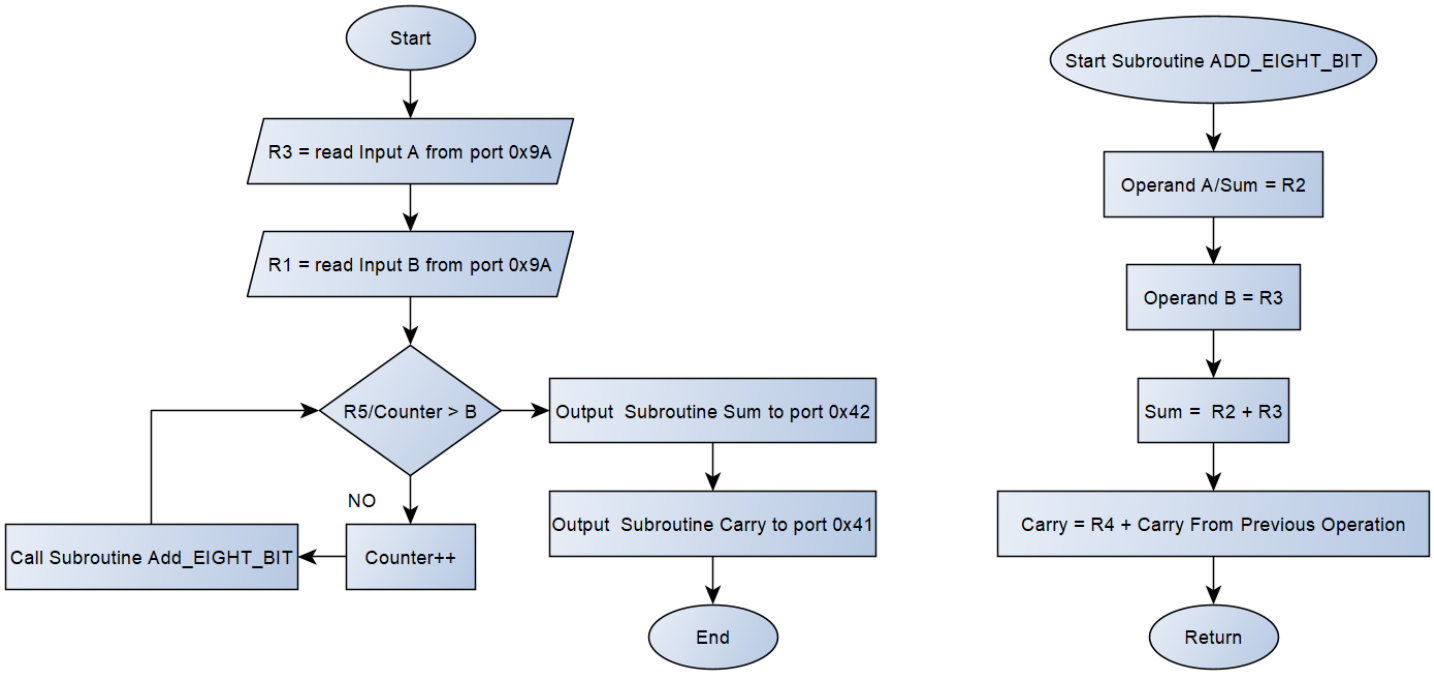
```
; Efficient 8bit to BCD converter
.EQU IN_PORT = 0x9A
.EQU OUT_PORT_1 = 0x43
.EQU OUT_PORT_10 = 0x42
.EQU OUT_PORT_100 = 0x41

.CSEG
.ORG 0x01
; Registers Used
; R0- Input
; R1- Subroutine Quotient
; R2- Subroutine Remainder
main:    IN R0, IN_PORT
         CALL DIV_10
         MOV R0, R1
         OUT R2, OUT_PORT_1
         CALL DIV_10
         OUT R2, OUT_PORT_10
         OUT R1, OUT_PORT_100
END: BRN END
; Subroutine DIV_10
; Description:
; Subroutine divides input value by 10
;
; Parameter Registers:
; R0- Input
; R1- Quotient
; R2- Remainder
DIV_10:  MOV R1, 0x00  ; Quotient/Counter
         MOV R2, R0        ; Remainder
DIFF:    CMP R2, 0x0A  ; Remainder < 10?
         BRCS END_SUB  ; if True, branch
         ADD R1, 0x01  ; Else, increment Quotient/Counter
         SUB R2, 0x0A  ; Remainder - 10
         BRN DIFF
END_SUB:RET
```

Behavior

Program 2: A Rat Assembly 16-bit multiplier. The program reads operands A & B from port 0x9A, and uses the ADD_EIGHT_BIT subroutine to compute the product. The subroutine adds the carry from each iteration to a Register storing the 'Top' 4-bits of the product. The program then outputs the 'Top' and 'Bottom' parts of the 16-bit product to ports 0x41 and 0x42 respectively.

Flowchart

Program 2



Verification

Program 2 Verification

Below we have provided four test trials of the 16b multiplier program. In some cases, the iterations can span many loops, which would be very impractical to tabulate.

Trial #	Input A (R3)	Input B (R1)	Top (R4)	Bot (R2)
1	0x00	0xFF	0x00	0x00
2	0xFF	0x00	0x00	0x00
3	0x55	0x1	0x00	0xFF
4	0x04	0x40	0x01	0x00

PROGRAM 2: 16b Multiplier

```
.EQU IN_PORT = 0x9A
.EQU OUT_PORT_TOP = 0x41
.EQU OUT_PORT_BOT = 0x42

.CSEG
.ORG 0x01
; Efficient Program that computes 16b product of
; 8 bit values, A * B
;
; Registers Used
; R1- Input B
; R2- Output Bot / Subroutine Operand A & Sum
; R3- Input A  / Subroutine Operand B
; R4- Output Top / Subroutine Carry
; R5- Counter
        MOV R2, 0x00  ; Initialize Registers
        MOV R4, 0x00
        MOV R5, 0x00
        IN R3, IN_PORT     ; Input A
        IN R1, IN_PORT     ; Input B
        CMP R3, 0x00
        BREQ OUTPUT
MULT:   CMP R5, R1         ; counter > B?
        BREQ OUTPUT
        ADD R5, 0x01  ; counter++
        CALL ADD_EIGHT_BIT
        BRN MULT
OUTPUT: OUT R2, OUT_PORT_BOT
        OUT  R4, OUT_PORT_TOP
END: BRN END

; Subroutine ADD_EIGHT_BIT
; Description:
; Subroutine adds two 8b registers and
; stores the carry.
;
; R2- Operand A & Sum
; R3- Operand B
; R4- Carry
ADD_EIGHT_BIT:
                ADD R2,R3     ; Sum = A + B
                ADDC R4, 0x00 ; Store Carry
END_SUB:    RET
```