LAB 3-D

ARITHMETIC INSTRUCTIONS

OBJECTIVES:

- To write a program to perform calculations.

MATERIAL:

- Atmel Studio

WEB SITES:

- www.microchip.com for Atmel Studio Software

ACTIVITY 1

Write a program that calculates (PORTC + 4) * PORTD and sends out the result through PORTB. Consider all the values are unsigned.

```
RESET:
    ; Configuration
    LDI R16, 0x00
    OUT DDRC, R16       ; Set PORTC as input
    OUT DDRD, R16       ; Set PORTD as input
    LDI R16, 0xFF
    OUT DDRB, R16       ; Set PORTB as output

MAIN:
    ; Read and Add
    IN  R16, PINC       ; Read value from PORTC
    LDI R17, 4          ; Load the constant 4
    ADD R16, R17        ; R16 = PORTC + 4

    ; Multiply
    IN  R18, PIND       ; Read value from PORTD
    MUL R16, R18        ; Multiply (R16 + 4) * PORTD
                        ; Result is stored in R1 (High byte) and R0 (Low byte)

    ;  Output
    OUT PORTB, R0       ; Send the lower byte of the result to PORTB

    RJMP MAIN           ; Loop back to keep reading inputs
```

LAB 3-D

ARITHMETIC INSTRUCTIONS

ACTIVITY 2

Write a program to calculate the result of (PORTB + PORTD)/2 and send out the result through PORTB. Consider all the values are unsigned. (Note: To divide by two, you can use shift right operation)

```
RESET:
    ; Configuration
    LDI R16, 0x00
    OUT DDRD, R16       ; Set PORTD as input

    ; Note: PORTB is used for both Input and Output
    OUT DDRB, R16       ; Set PORTB as input initially to read value

MAIN:
    ; Read Inputs
    IN  R16, PINB       ; Read value from PORTB
    IN  R17, PIND       ; Read value from PORTD

    ; Addition
    ADD R16, R17        ; R16 = PORTB + PORTD

    ;  Division by 2
    ;  use Logical Shift Right (LSR)
    LSR R16             ; Shift bits right once to divide by 2

    ; Output Result
    LDI R18, 0xFF
    OUT DDRB, R18       ; Switch PORTB to Output mode
    OUT PORTB, R16      ; Send result out through PORTB

    ; Reset DDRB
    LDI R18, 0x00
    OUT DDRB, R18

    RJMP MAIN           ; Loop back to keep reading inputs
```

LAB 3-D

ARITHMETIC INSTRUCTIONS

ACTIVITY 3

1) Find the value in R0 and R1 after the following code. What are the values kept in R0 and R1?

```
        LDI   R16, 10 ; Load 10 into R16
        LDI   R17, 20 ; Load 20 into R17
        LDI   R18, 30 ; Load 30 into R18
        MUL   R16, R17 ; Multiply R16 * R17 (10 * 20 = 200)
        ADD   R0, R18 ; Add R18 to the result in R0 (200 + 30)
```

Final Values: **R0 = 230 R1 = 0**

2) Find the value in R0 and R1 after the following code.What are the values kept in R0 and R1?

```
        LDI   R19, 19 ; Load 19 into R19
        SUBI  R19, 10 ; Subtract 10 from R19 (19 - 10 = 9)
        LDI   R30, 30 ; Load 30 into R30
        MUL   R30, R1 ; Multiply R30 * R19 (30 * 9 = 270)
```

Final Values: **R0 = 14 R1 = 1**

AVR hardware automatically splits multiplication results into R1 (High Byte) and R0 (Low Byte) to accommodate values larger than 255. For your $30 \times 9 = 270$ example, 270 is too big for one register, so R1 stores 1 (worth 256) and R0 stores 14, totaling 270

LAB 3-D

ARITHMETIC INSTRUCTIONS

ACTIVITY 4

Write a program to add 10 bytes of data and store the result in registers R30 and R31. The bytes are stored in the **Program memory** starting at $200. The data would look as follows:

```
MYDATA: .DB     92,34,84,129,...        ;pick your own data.
```

Note you must first bring the data from Program memory into the registers, then add them together. Use a simulator and single-step to examine the data.

```
.ORG $0000
    RJMP RESET

; Define my own data at address $200
.ORG $0200
MYDATA: .DB 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

RESET:
    ; Initialize Z pointer
    LDI ZL, LOW(MYDATA << 1)
    LDI ZH, HIGH(MYDATA << 1)

    LDI R20, 10     ; Loop counter for 10 bytes
    CLR R30         ; Clear Low Byte of result
    CLR R31         ; Clear High Byte of result
    CLR R1          ; Ensure R1 is 0 for the ADC instruction

ADD_LOOP:
    ; Bring data from Program Memory to registers
    LPM R16, Z+     ; Load byte into R16 and move to next address

    ;Add them together into R31:R30
    ADD R30, R16    ; Add new byte to Low Byte
    ADC R31, R1     ; Add the carry bit to the High Byte

    DEC R20         ; Decrease counter
    BRNE ADD_LOOP   ; Continue until all 10 bytes are added

DONE:
    RJMP DONE
```

R15	0x00
R16	0xFF
R17	0x00
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0x00
R23	0x00
R24	0x00
R25	0x00
R26	0x00
R27	0x00
R28	0x00
R29	0x00
R30	0x42
R31	0x08

LAB 3-D

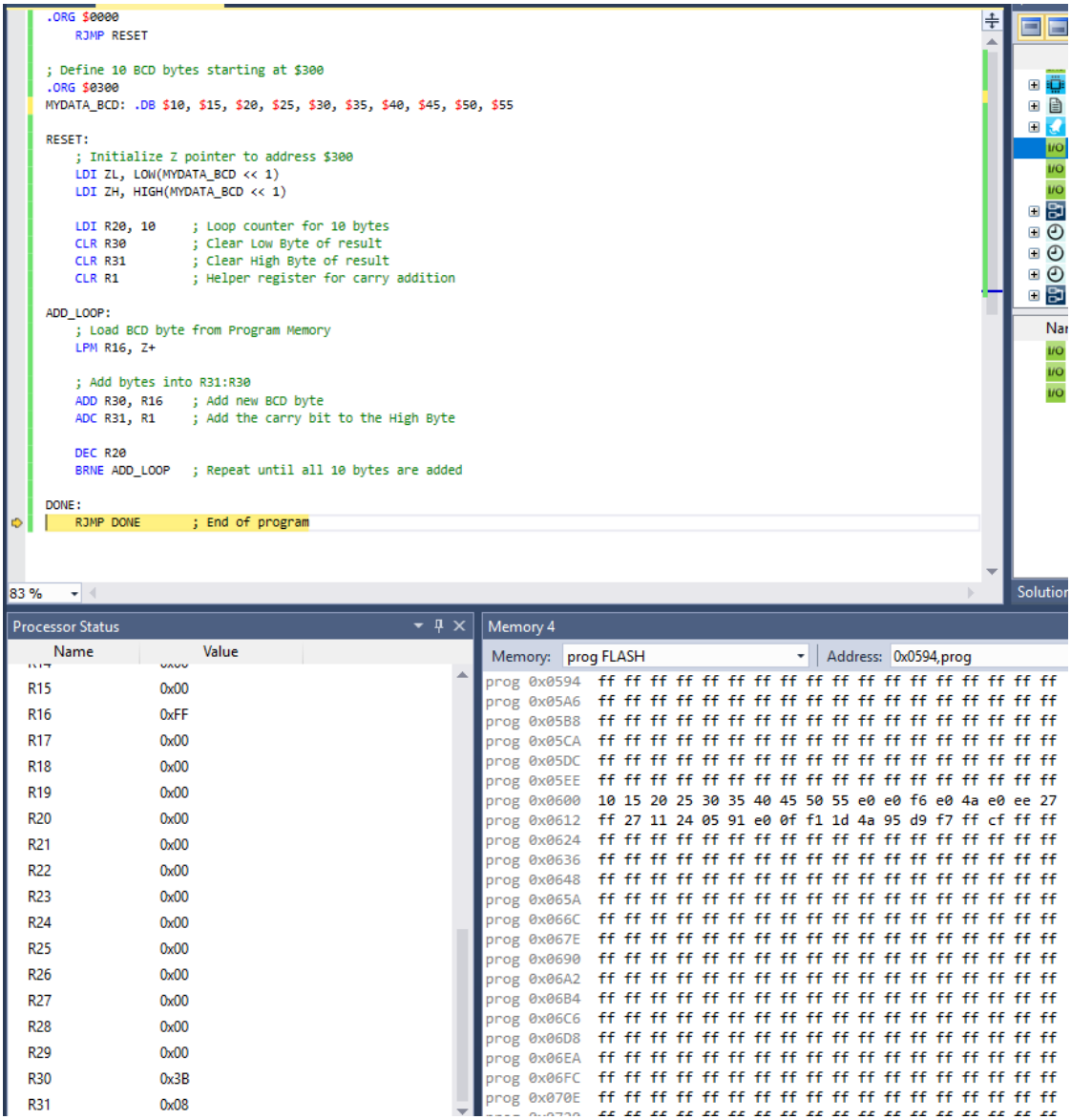
ARITHMETIC INSTRUCTIONS

ACTIVITY 5

Write a program to add 10 bytes of Binary-Coded Decimal (BCD) data and store the result in R30 and R31. The bytes are stored in **Program memory** starting at $300. The data would look as follows:

```
MYDATA: .DB     $92,$34,$84,$29,...     ;pick your own data.
```

Note you must first bring the data from Program memory into the registers, then add them together. Use a simulator and single-step to examine the data.



```
.ORG $0000
    RJMP RESET

; Define 10 BCD bytes starting at $300
.ORG $0300
MYDATA_BCD: .DB $10, $15, $20, $25, $30, $35, $40, $45, $50, $55

RESET:
    ; Initialize Z pointer to address $300
    LDI ZL, LOW(MYDATA_BCD << 1)
    LDI ZH, HIGH(MYDATA_BCD << 1)

    LDI R20, 10     ; Loop counter for 10 bytes
    CLR R30         ; Clear Low Byte of result
    CLR R31         ; Clear High Byte of result
    CLR R1          ; Helper register for carry addition

ADD_LOOP:
    ; Load BCD byte from Program Memory
    LPM R16, Z+

    ; Add bytes into R31:R30
    ADD R30, R16    ; Add new BCD byte
    ADC R31, R1     ; Add the carry bit to the High Byte

    DEC R20
    BRNE ADD_LOOP   ; Repeat until all 10 bytes are added

DONE:
    RJMP DONE       ; End of program
```

83 %

Processor Status

Name	Value
R15	0x00
R16	0xFF
R17	0x00
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0x00
R23	0x00
R24	0x00
R25	0x00
R26	0x00
R27	0x00
R28	0x00
R29	0x00
R30	0x3B
R31	0x08

Memory 4

Memory: prog FLASH Address: 0x0594,prog

```
prog 0x0594  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x05A6  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x05B8  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x05CA  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x05DC  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x05EE  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x0600  10 15 20 25 30 35 40 45 50 55 e0 e0 f6 e0 4a e0 ee 27
prog 0x0612  ff 27 11 24 05 91 e0 0f f1 1d 4a 95 d9 f7 ff cf ff ff
prog 0x0624  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x0636  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x0648  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x065A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x066C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x067E  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x0690  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x06A2  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x06B4  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x06C6  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x06D8  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x06EA  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x06FC  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
prog 0x070E  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
```

LAB 3-D

ARITHMETIC INSTRUCTIONS

ACTIVITY 6

Write a program to add two BCD numbers and store the result in **RAM** location $100 - $104. The two multibyte items are stored in the program memory starting at $120 as following data.

```
          .ORG   $120
DATA_1:   .DB    $54,$76,$65,$98    ;number 0x98657654
DATA_2:   .DB    $93,$56,$77,$38    ;number 0x38775693
```

```
.ORG $0000
RJMP MAIN

;--- Program Memory Data ---
.ORG $0120
DATA_1: .DB $54, $76, $65, $98 ; 0x98657654 (Stored LSB first)
DATA_2: .DB $93, $56, $77, $38 ; 0x38775693 (Stored LSB first)

MAIN:
    ; 1. Setup Result Pointer (X = R27:R26) to RAM $100
    LDI R26, $00
    LDI R27, $01

    ; 2. Setup Z and Y pointers for the data arrays
    LDI ZL, LOW(DATA_1 << 1)
    LDI ZH, HIGH(DATA_1 << 1)
    LDI YL, LOW(DATA_2 << 1)
    LDI YH, HIGH(DATA_2 << 1)

    ; 3. Initialize Loop Counter (4 bytes)
    LDI R20, 4
    CLC ; Clear Carry initially

LOOP:
    ; 4. Load bytes using auto-increment pointers
    LPM R18, Z+ ; Load byte from DATA_1 and move Z to next
    LPM R19, Y+ ; Load byte from DATA_2 and move Y to next

    ; 5. Add with Carry (ADC)
    ADC R18, R19

    ; 6. Store result in RAM $100-$103 and increment X
    ST X+, R18

    DEC R20
    BRNE LOOP

    ; 7. Store final Carry in $104
    CLR R19
    CLR R18
    ADC R18, R19 ; Correctly grab the final carry bit
    ST X, R18    ; Store in RAM $104

HERE:
    RJMP HERE
```