LAB 4-B

DATA TRANSFER

OBJECTIVES:

- ⍰ To code a program to transfer data from program memory into RAM locations.
- ⍰ To code a program to transfer data from RAM locations to other RAM locations.
- ⍰ To experiment with a look-up table.

MATERIAL:

- ⍰ Atmel Studio

WEB SITES:

- ⍰ www.microchip.com for Atmel Studio Software

ACTIVITY 1

Write a program to transfer a string of data from **program memory** starting at address $200 to RAM locations starting at $140. Using the simulator, single-step through the program and examine the data transfer and registers.

```
.INCLUDE "m328pdef.inc"

.CSEG
.ORG 0x200
MY_DATA: .DB "AVR LAB 4B", 0  ; String with null terminator (0)

.ORG 0x0000
    ; 1. Initialize Stack Pointer (Good practice)
    LDI R16, LOW(RAMEND)
    OUT SPL, R16
    LDI R16, HIGH(RAMEND)
    OUT SPH, R16

    ; 2. Set Z Pointer to Program Memory Source (Byte Address)
    ; Flash is word-addressed, so multiply by 2 for byte address
    LDI ZL, LOW(2 * MY_DATA)
    LDI ZH, HIGH(2 * MY_DATA)

    ; 3. Set Y Pointer to SRAM Destination (0x0140)
    LDI YL, 0x40
    LDI YH, 0x01
```

LAB 4-B

DATA TRANSFER

```
COPY_LOOP:
    LPM R16, Z+         ; Load byte from Flash into R16, increment Z
    CPI R16, 0        ; Check for null terminator (end of string)
    BREQ DONE           ; If 0, finish

    ST Y+, R16          ; Store byte to SRAM pointed by Y, increment Y
    RJMP COPY_LOOP      ; Repeat

DONE:
    RJMP DONE
```

ACTIVITY 2

Add the subroutine to the program in Activity 1. After data has been transferred from program memory into RAM, the subroutine function should copy the data from **RAM** locations starting at $140 to **RAM** locations starting at $160. Use single-step through the subroutine and examine the operations.

```
.INCLUDE "m328pdef.inc"

.DSEG
.ORG 0x0140
    ; (Data is expected to be here from Activity 1)

.CSEG
.ORG 0x0000
    ; Initialize Stack Pointer (REQUIRED for Subroutines)
    LDI R16, LOW(RAMEND)
    OUT SPL, R16
    LDI R16, HIGH(RAMEND)
    OUT SPH, R16

    ; ... [Include Activity 1 Code here to populate RAM first] ...

    ; Call the RAM-to-RAM copy subroutine
    RCALL COPY_RAM_TO_RAM

STOP: RJMP STOP

; --------------------------------------
; Subroutine: Copy 10 bytes from 0x140 to 0x160
; --------------------------------------
COPY_RAM_TO_RAM:
    PUSH R16            ; Save registers used
    PUSH R17
    PUSH XL
    PUSH XH
    PUSH YL
    PUSH YH

    ; Set X Pointer to Source (0x0140)
    LDI XL, 0x40
    LDI XH, 0x01

    ; Set Y Pointer to Destination (0x0160)
    LDI YL, 0x60
    LDI YH, 0x01

    LDI R17, 10        ; Counter (assuming string length 10)

RAM_LOOP:
    LD R16, X+         ; Load from Source (RAM)
    ST Y+, R16         ; Store to Destination (RAM)
    DEC R17
    BRNE RAM_LOOP

    POP YH             ; Restore registers
    POP YL
    POP XH
    POP XL
    POP R17
    POP R16
    RET
```

ACTIVITY 3

1. Write a program to calculate y where y = $x^2 + 2x + 9$. Where x is the number between 0 and 9 and the look-up table for x^2 is located at the address \$200 of **program memory**. Register R20 keeps the value of x, and at the end of the program R21 should contain the value of y. Use the simulator to change the x value and single-step through the program, examining the changes.

```
.INCLUDE "m328pdef.inc"

.CSEG
.ORG 0x200
SQUARE_TABLE:
    .DB 0, 1, 4, 9, 16, 25, 36, 49, 64, 81  ; x^2 values for x=0 to 9
```

Lab 4-B

Data Transfer

```
.ORG 0x0000
    ; Define X (Example: x = 5)
    LDI R20, 5

    ; 1. Get x^2 from Look-up Table
    LDI ZL, LOW(2 * SQUARE_TABLE)
    LDI ZH, HIGH(2 * SQUARE_TABLE)

    ; Add Offset x (R20) to Z Pointer
    ADD ZL, R20
    LDI R16, 0
    ADC ZH, R16         ; Add carry to high byte if needed

    LPM R21, Z        ; Load x^2 into R21 (R21 = 25 for x=5)

    ; 2. Calculate 2x
    MOV R22, R20        ; Copy x to R22
    LSL R22           ; Logic Shift Left (Multiply by 2). R22 = 10.

    ; 3. Calculate x^2 + 2x
    ADD R21, R22        ; R21 = 25 + 10 = 35

    ; 4. Add 9
    LDI R23, 9
    ADD R21, R23        ; R21 = 35 + 9 = 44 (Result y)

HERE: RJMP HERE
```

2. Explain the difference between the following two instructions:
   a. LPM R16, Z
   b. LD R16, Z

   ◦ **LPM** (Load Program Memory): Loads one byte of data from Flash Memory (Program Memory) pointed to by the Z register into a general-purpose register.
   ◦ **LD** (Load Indirect): Loads one byte of data from SRAM (Data Memory) pointed to by the Z register (or X/Y) into a general-purpose register.

3. Circle the invalid instructions.
   a. LDS R20, 60. -> Valid.

   b. LD R30, Z -> Valid.

   c. LD R25, Z+ -> Valid.

Lab 4-B

Data Transfer

d. LPM R25, Z+4 -> INVALID because the AVR instruction set does not support adding an immediate offset (displacement) directly to the Z pointer in the LPM instruction. It only supports LPM Rd, Z or LPM Rd, Z+.

4. Explain the difference between the following two instructions:

   a. LDS R20, $40

   b. LDI R20, $40

   ◦ **LDS R20, $40** (Load Direct from Data Space): Reads the contents stored at memory address $0040 in SRAM and copies that data into register R20.
   ◦ **LDI R20, $40** (Load Immediate): Loads the number/value $40 (decimal 64) directly into register R20.