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"
.ORG $0000
   RJMP MAIN

.ORG $0200
MY_STR: .DB "ASSEMBLY", 0  ; String ends with a null terminator
.ORG $0010

MAIN:
   LDI ZL, LOW(MY_STR << 1)
   LDI ZH, HIGH(MY_STR << 1)
   LDI XL, $40
```

```
    LDI XH, $01

COPY_LOOP:
    ; 3. Transfer Data
    LPM R16, Z+     ; Load from Flash, increment Z
    ST  X+, R16     ; Store to RAM, increment X
    CPI R16, 0
    BRNE COPY_LOOP

HERE: RJMP HERE
```

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
 RCALL COPY_RAM_TO_RAM
STOP: RJMP STOP


COPY_RAM_TO_RAM:
 PUSH R16 ; Save registers used
LAB 4-B
DATA TRANSFER
 PUSH R17
 PUSH XL
 PUSH XH
 PUSH YL
 PUSH YH


 LDI XL, 0x40
 LDI XH, 0x01


 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"

.ORG 0x0000

    RJMP MAIN


; Look-up table for x^2 (0^2 to 9^2) at $200

.ORG 0x0200

SQUARE_TABLE: .DB 0, 1, 4, 9, 16, 25, 36, 49, 64, 81


.ORG 0x0010

MAIN:

    LDI R16, LOW(RAMEND)

    OUT SPL, R16

    LDI R16, HIGH(RAMEND)

    OUT SPH, R16
```

LAB 4-B

DATA TRANSFER

```
    LDI R20, 5          ; Example: x = 5

    LDI ZL, LOW(SQUARE_TABLE << 1)
    LDI ZH, HIGH(SQUARE_TABLE << 1)

    CLR R16             ; Clear R16 to use for Z-offset
    ADD ZL, R20           ; Add x to Z-pointer low byte
    ADC ZH, R16           ; Add carry to high byte

    LPM R21, Z           ; R21 = x^2 (fetched from table)

    MOV R18, R20          ; Copy x to R18
    LSL R18              ; Logical Shift Left: R18 = 2x
    ADD R21, R18          ; R21 = x^2 + 2x

    LDI R18, 9
    ADD R21, R18          ; R21 = x^2 + 2x + 9

  HERE: RJMP HERE
```

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

LPM R16, Z (Load Program Memory): Fetches data from Flash memory (where your code is stored). It is used for constant data, like lookup tables or strings.

Lab 4-B

Data Transfer

LD R16, Z (Load Indirect): Fetches data from SRAM (Data memory). It is used for variables and data that change during program execution.

3. Circle the invalid instructions.
   a. LDS R20, 60

   b. LD R30, Z

   c. LD R25, Z+

   d. LPM R25, Z+4

Invalid, AVR instruction set does not support adding displacement to the Z pointer in LPM instruction directly. It only support LPM Rd, Z or LPM Rd, Z+

4. Explain the difference between the following two instructions:
   a. LDS R20, $40

   b. LDI R20, $40

A. LDS R20, $40 (Load Direct from Data Space): This treats $40 as a memory address. It goes to SRAM location $40, looks at the value stored there, and copies it into R20.

B. LDI R20, $40 (Load Immediate): This treats $40 as a constant value. It simply puts the number 64 (hex $40) directly into R20.