Class 6 – AVR Addressing Modes

Overview:

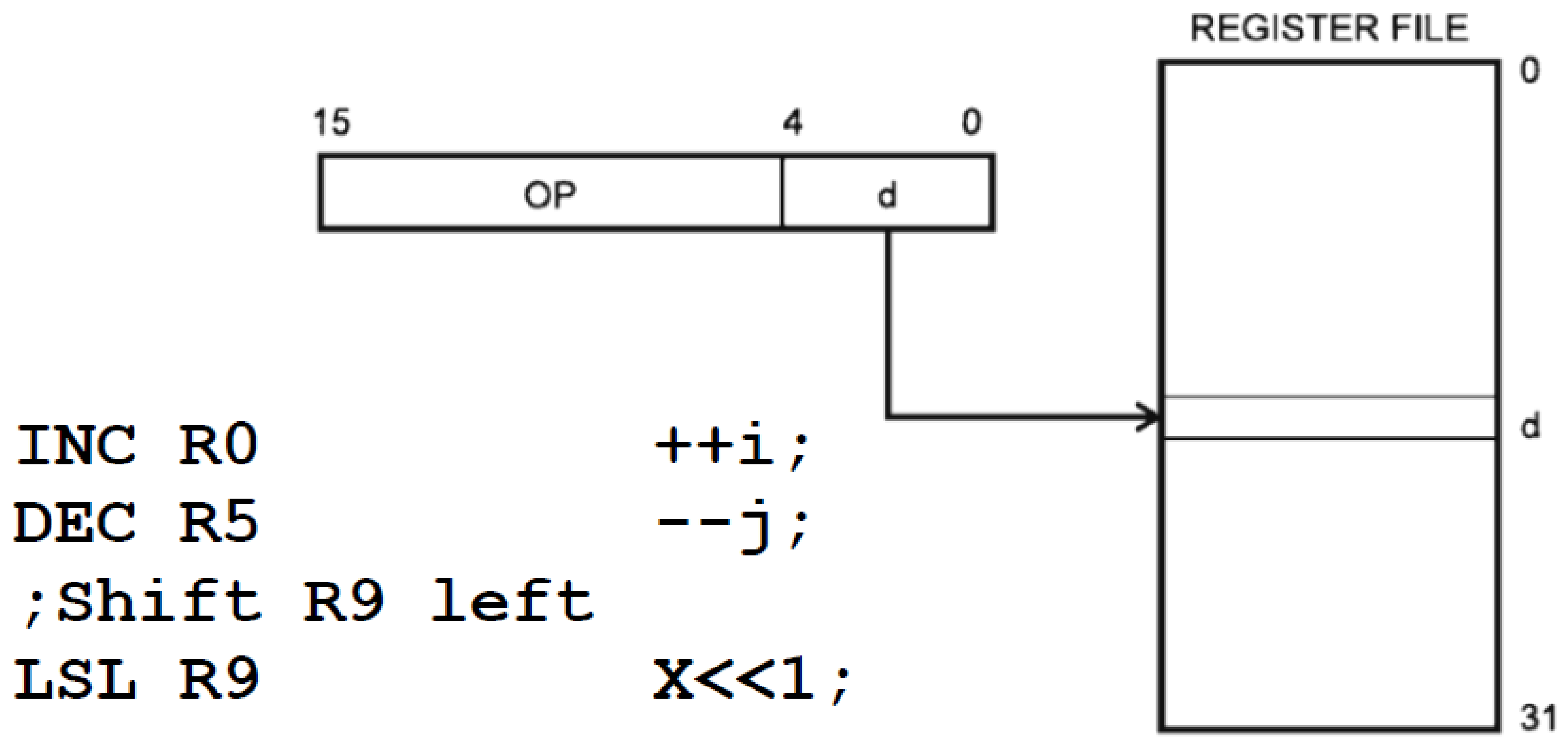
- The Instruction set of a processor is like the vocabulary of the processor.
- Each instruction controls some part of the processor and allows the programmer to manipulate data in the memory as well as input and output devices.

Program and Data Addressing Modes:

- The instructions can be categorized based on how they access data and operate upon it. This is called the program and data addressing modes of the processor.
- The various AVR instructions can be categorized in about 10 different addressing modes.
- Each instruction typically has an 2 parts:
  - Opcode: indicates to ALU what to do.
  - Operands: on which the opcode operates. (which may be implicit in the instruction itself…)

1- Register Direct (Single Register)

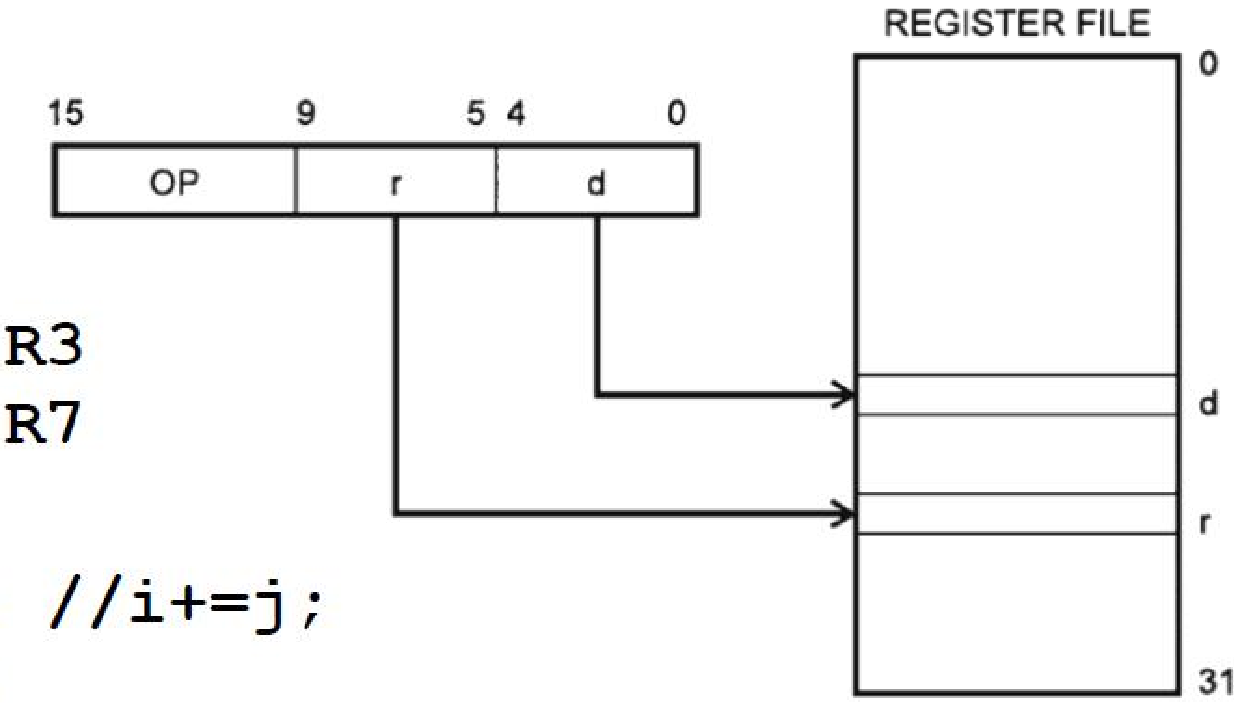
- Register Direct instructions can operate on any of the 32 registers.
- Instruction Reads the contents of a register, operates on the contents, and then stores the result back into the same register.



```
INC R0              ++i;
DEC R5              --j;
;Shift R9 left
LSL R9              X<<1;
```

2 - Register Direct (Two Registers)

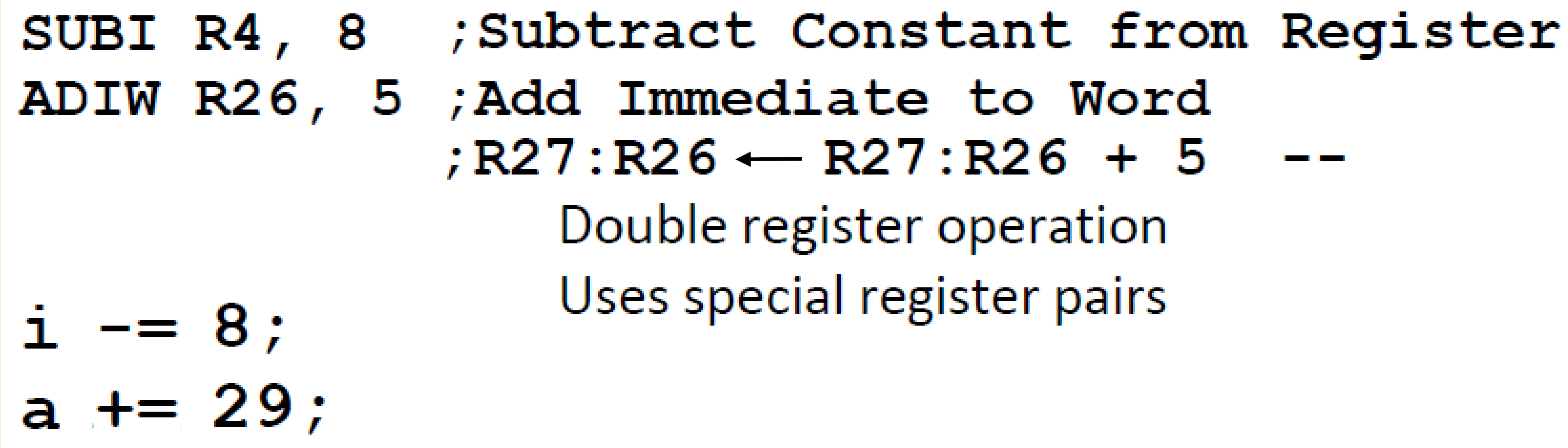
- Two registers are involved.
  - Rs: Source register
  - Rd: Destination register
  - Instruction reads the two registers and operates on their contents and stores the result back in the destination register.



```
ADD R1, R3
SUB R5, R7

i = i+j; //i+=j;
i = i-j;
```

3 – Immediate Mode

- A constant value is in the instruction
- This will be store with program code in program memory space (Flash memory on AVR)



```
SUBI R4, 8   ;Subtract Constant from Register
ADIW R26, 5  ;Add Immediate to Word
             ;R27:R26 ← R27:R26 + 5   --
```

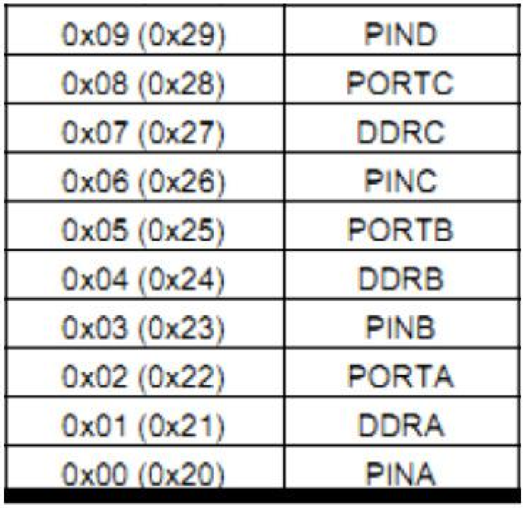
Double register operation

Uses special register pairs

```
i -= 8;
a += 29;
```

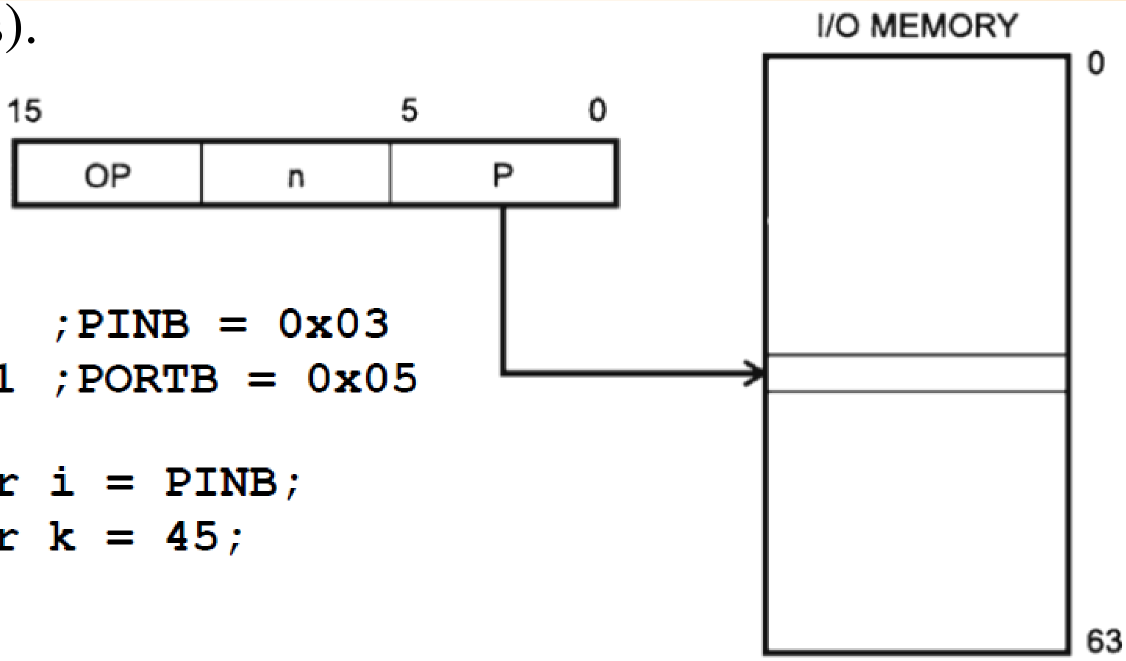
4 – I/O Direct

- These instructions are used to access the I/O space. (I/O Registers) [Not Extended I/O Registers!!!]
- The I/O registers can be accessed using
  - In Rd, PORTADDRESS
  - Out PORTADDRESS, Rs
  - Rd, Rs: any register from register file.
- PORTADDRESS : any register from the entire range of 0x00 to 0x3F (not 0x20-0x5F) (a total of 64 I/O registers).



0x09 (0x29)	PIND
0x08 (0x28)	PORTC
0x07 (0x27)	DDRC
0x06 (0x26)	PINC
0x05 (0x25)	PORTB
0x04 (0x24)	DDRB
0x03 (0x23)	PINB
0x02 (0x22)	PORTA
0x01 (0x21)	DDRA
0x00 (0x20)	PINA

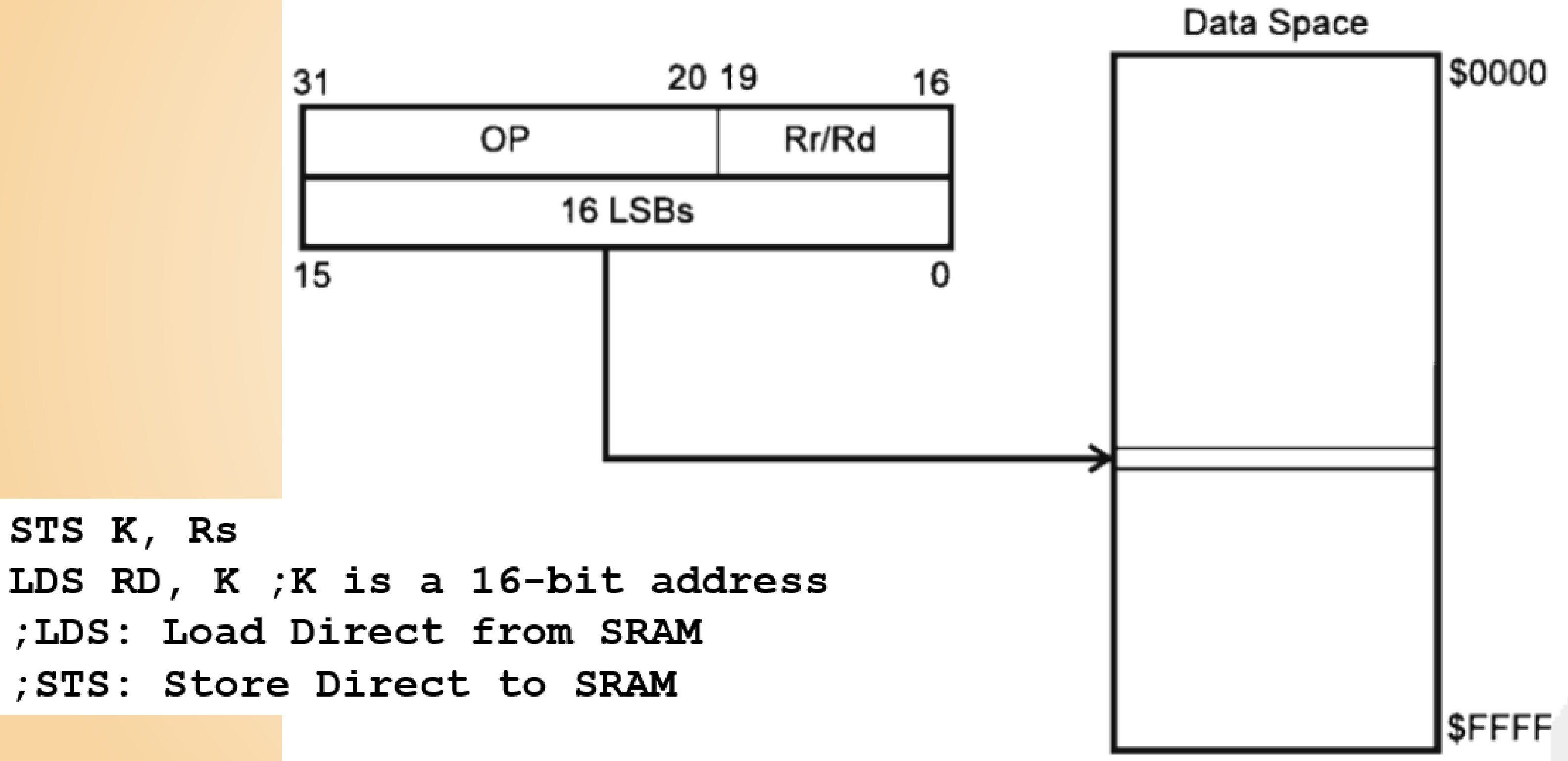
*page 376 of data sheet



```
IN R10, PINB   ;PINB = 0x03
OUT PORTB, R1  ;PORTB = 0x05

unsigned char i = PINB;
unsigned char k = 45;
PORTB = k;
```

5 – Data Direct

- These are two-word instructions
- One of the words is the address of the data memory space.
    - What is the maximum data memory that can be accessed in this way?



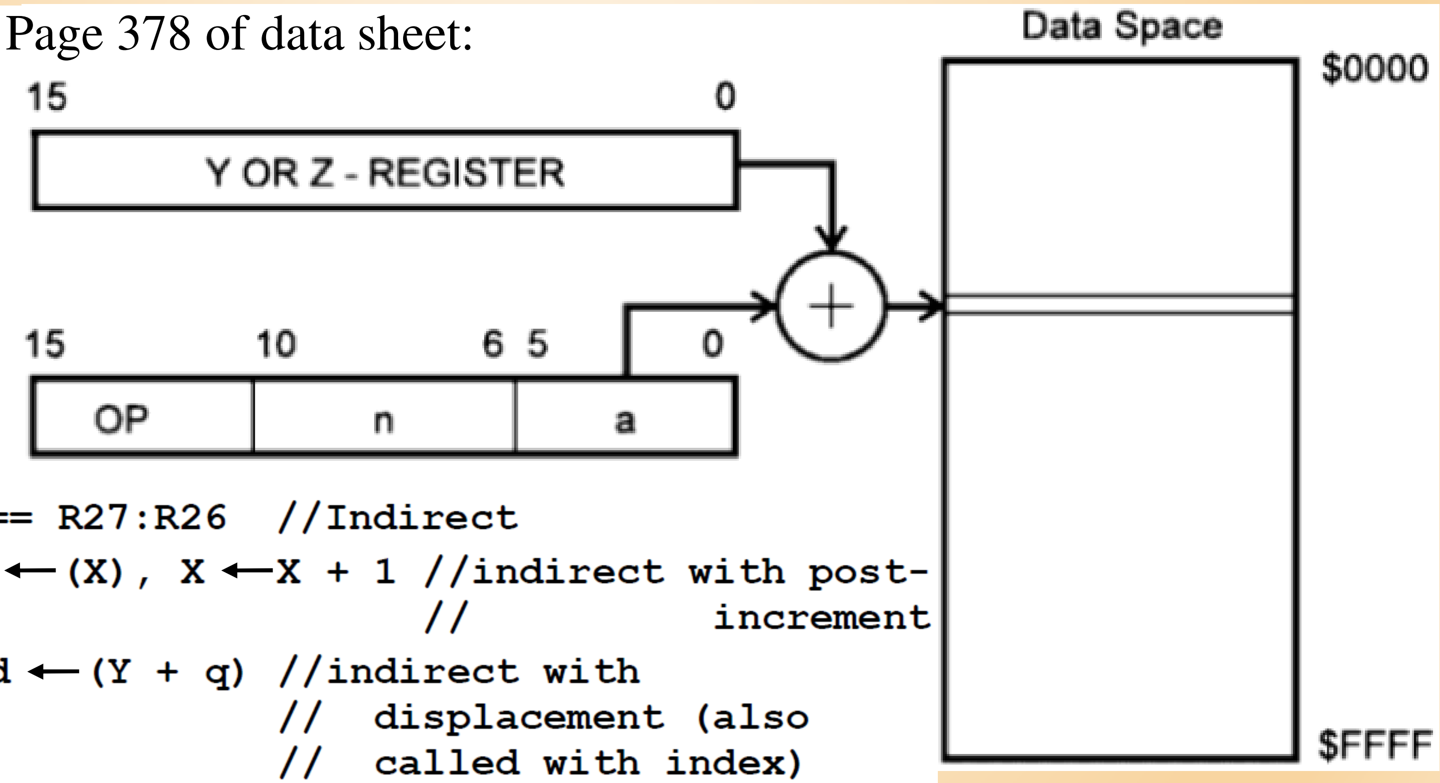
```
STS K, Rs
LDS RD, K ;K is a 16-bit address
;LDS: Load Direct from SRAM
;STS: Store Direct to SRAM
```

- Note: - writing a constant to memory requires writing it to a register first.

6 – Data Indirect

- Similar to the data direct type.
- But they are one word each, and a pointerregister (X, Y, or Z)is used that has the base address of the data memory.
- This figure is one variant

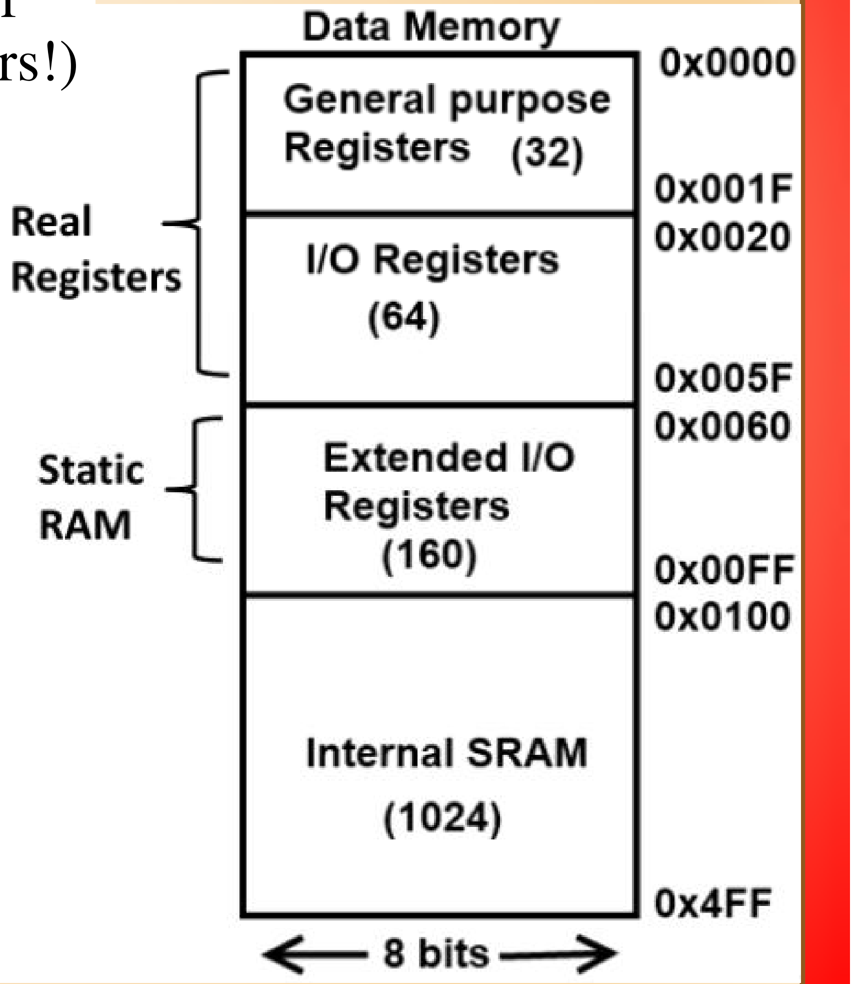
Look at Page 378 of data sheet:



```
LD Rd, X    ;X == R27:R26   //Indirect
LD Rd, X+   ;Rd ←(X), X ←X + 1 //indirect with post-
                               //           increment
LDD Rd, Y+q ;Rd ←(Y + q) //indirect with
                         //  displacement (also
                         //  called with index)
ST  -Y, Rs ;Y ←Y - 1, (Y) ← Rs //indirect with pre-
                               //decrement
```

6 – (cont) I/O Ports using Indirect

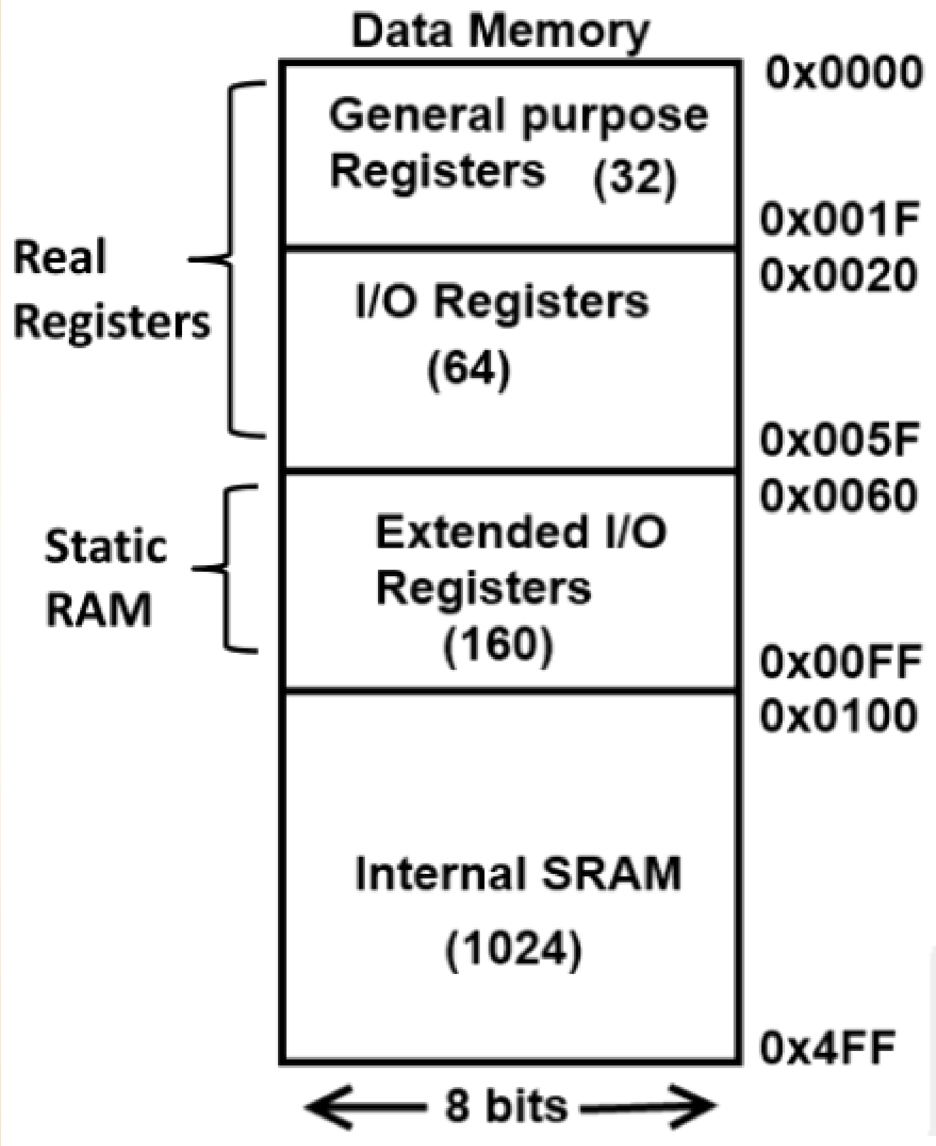
- The ports can also be accessed using SRAM access commands, e.g. ST and LD. Just add 0x20 to the port number (the first 32 addresses are the registers!) and access the port that way. Like demonstrated here:



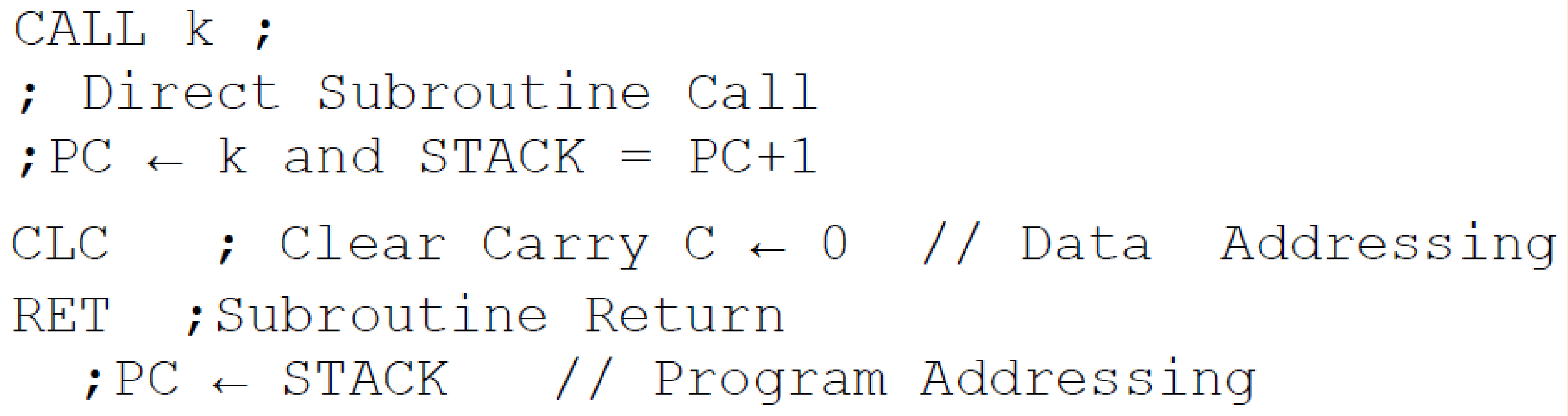
```
.DEF SomeRegister = R16

     LDI ZH,HIGH(PORTB+32)
     LDI ZL,LOW(PORTB+32)
     LD SomeRegister,Z
```

Extended I/O

- For I/O registers located in extended I/O map, I/O register direct commands like "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" cannot be used. They must be replaced with direct (memory) and indirect (memory) instructions that allow access to extended I/O, typically "LDS" and "STS" combined with "SBRS", "SBRC", "SBR", and "CBR".



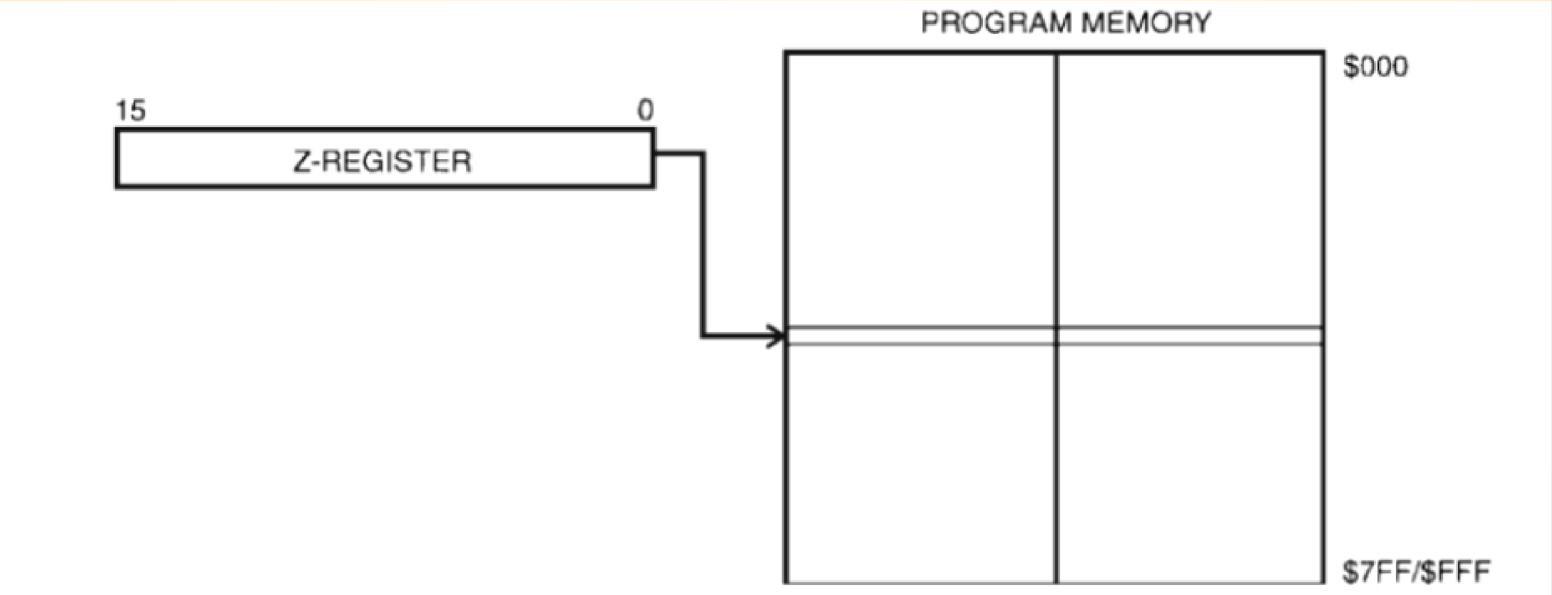
7 – Direct Program Addressing (8 – Implicit Addressing)



```
CALL k ;
; Direct Subroutine Call
;PC ← k and STACK = PC+1
CLC    ; Clear Carry C ← 0  // Data  Addressing
RET   ;Subroutine Return
   ;PC ← STACK    // Program Addressing
```

9 – Indirect Program Addressing

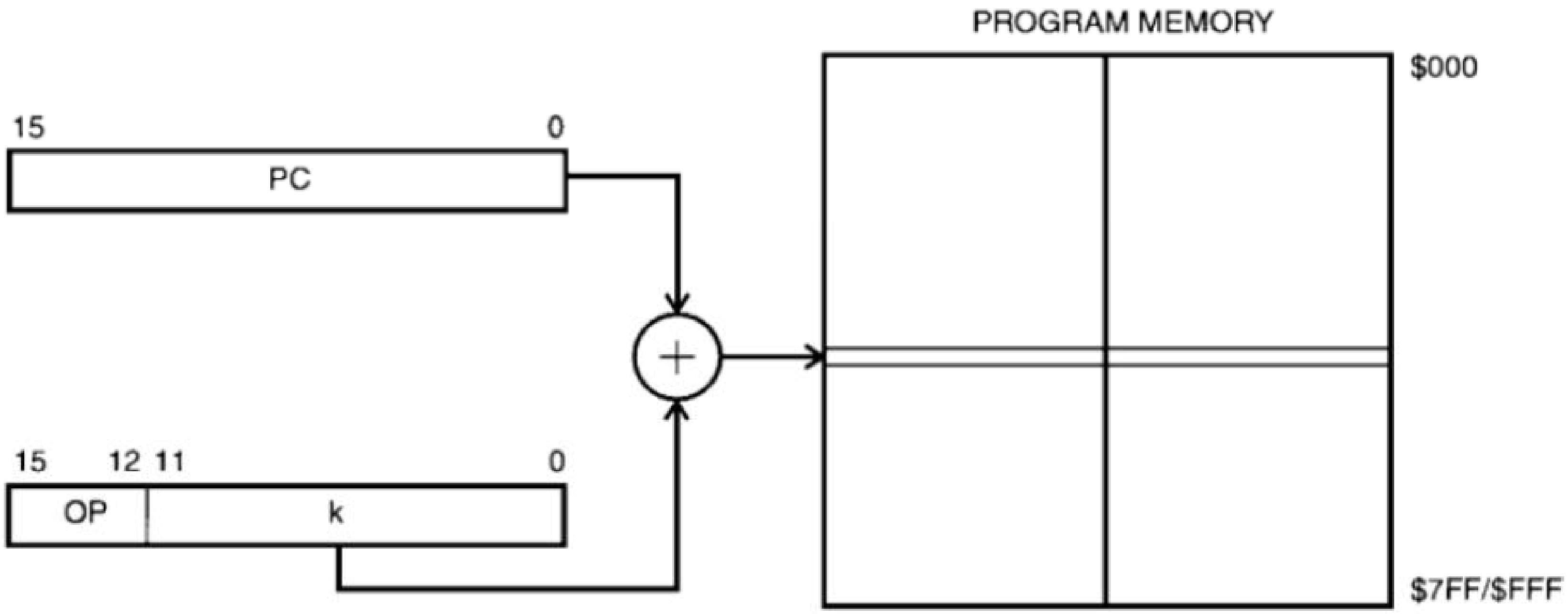
- In these types of instructions, the Z register is used to point to the program memory. (Up to 64 Kbytes of program memory)



```
IJMP  ;Indirect Jump to (Z) PC ← Z
ICALL ;Indirect Call to (Z) PC ← Z,STACK = PC  + 1
```

10 – Relative Program Addressing

- These instructions are of the type **RJMP** and **RCALL**, where an offset of +/-2K to the program counter is used.



RCALL k ;Relative Subroutine Call PC ← PC + k + 1 , STACK=PC+1

RJMP k ;Relative Jump PC ← PC + k + 1