Pointer - registers

A very special extra role is defined for the register pairs R26:R27, R28:R29 and R30:R31. The role is so important that these pairs have extra names in assembler: X, Y and Z. These pairs are 16-bit pointer registers, able to point to addresses with max. 16-bit into SRAM locations (X, Y or Z) or into locations in program memory (Z).

The lower byte of the 16-bit-adress is located in the lower register, the higher byte in the upper register. Both parts have their own names, e.g. the higher byte of Z is named ZH (=R31), the lower Byte is ZL (=R30). These names are defined in the standard header file for the chips. Dividing these 16-bit-pointer-names into two different bytes is done like follows:

```
.EQU Address = RAMEND ; RAMEND is the highest 16-bit address in SRAM

  LDI YH,HIGH(Address) ; Set the MSB

  LDI YL,LOW(Address) ; Set the LSB
```

Accesses via pointers are programmed with specially designed commands. Read access is named LD (LoaD), write access named ST (STore), e.g. with the X-pointer:

Pointer	Sequence	Example
X	Read/Write from address X, don't change the pointer	LD R1,X ST X,R1
X+	Read/Write from/to address X and increment the pointer afterwards by one	LD R1,X+ ST X+,R1
-X	Decrement the pointer by one and read/write from/to the new address afterwards	LD R1,-X ST -X,R1

Similarly you can use Y and Z for that purpose.

There is only one command for the read access to the program storage. It is defined for the pointer pair Z and it is named LPM (Load from Program Memory). The command copies the byte at address Z in the program memory to the register R0. As the program memory is organized word-wise (one command on one address consists of 16 bits or two bytes or one word) the least significant bit selects the lower or higher byte (0=lower byte, 1= higher byte). Because of this the original address must be multiplied by 2 and access is limited to 15-bit or 32 kB program memory. Like this:

```
LDI ZH,HIGH(2*Adress)

LDI ZL,LOW(2*Adress)

LPM
```

Following this command the address must be incremented to point to the next byte in program memory. As this is used very often a special pointer incrementation command has been defined to do this:

```
ADIW ZL,1

LPM
```

ADIW means ADd Immediate Word and a maximum of 63 can be added this way. Note that the assembler expects the lower of the pointer register pair ZL as first parameter. This is somewhat confusing as addition is done as 16-bit-operation.

The complement command, subtracting a constant value of between 0 and 63 from a 16-bit pointer register is named SBIW, Subtract Immediate Word. (SuBtract Immediate Word). ADIW and SBIW are possible for the pointer register pairs X, Y and Z and for the register pair R25:R24, that does not have an extra name and does not allow access to SRAM or program memory locations. R25:R24 is ideal for handling 16-bit values.

As incrementation after reading is very often needed, newer AVR types have the instruction

LPM R,Z+

This allows to transport the byte read to any location R, and auto-increments the pointer register.

How to insert that table of values in the program memory? This is done with the assembler directives .DB and .DW. With that you can insert bytewise or wordwise lists of values. Bytewise organized lists look like this:

.DB 123,45,67,89 ; a list of four bytes

.DB "This is a text. " ; a list of byte characters

You should always place an even number of bytes on each single line. Otherwise the assembler will add a zero byte at the end, which might be unwanted.

The similar list of words looks like this:

.DW 12345,6789 ; a list of two words

Recommendation for the use of registers

Define names for registers with the .DEF directive, never use them with their direct name Rx.

If you need pointer access reserve R26 to R31 for that purpose.

16-bit-counter are best located R25:R24.

If you need to read from the program memory, e.g. fixed tables, reserve Z (R31:R30) and R0 for that purpose.

If you plan to have access to single bits within certain registers (e.g. for testing flags), use R16 to R23 for that purpose.