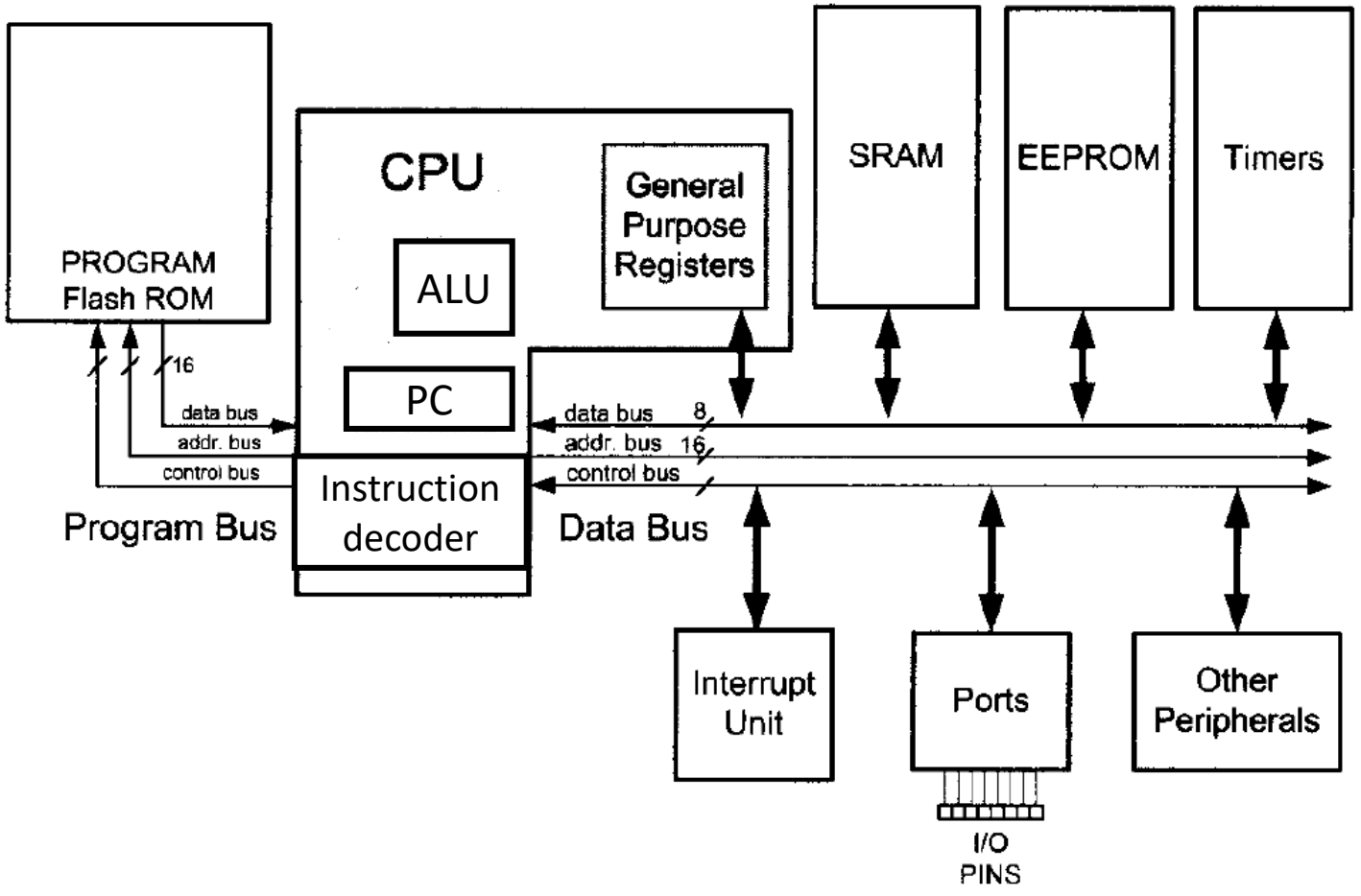
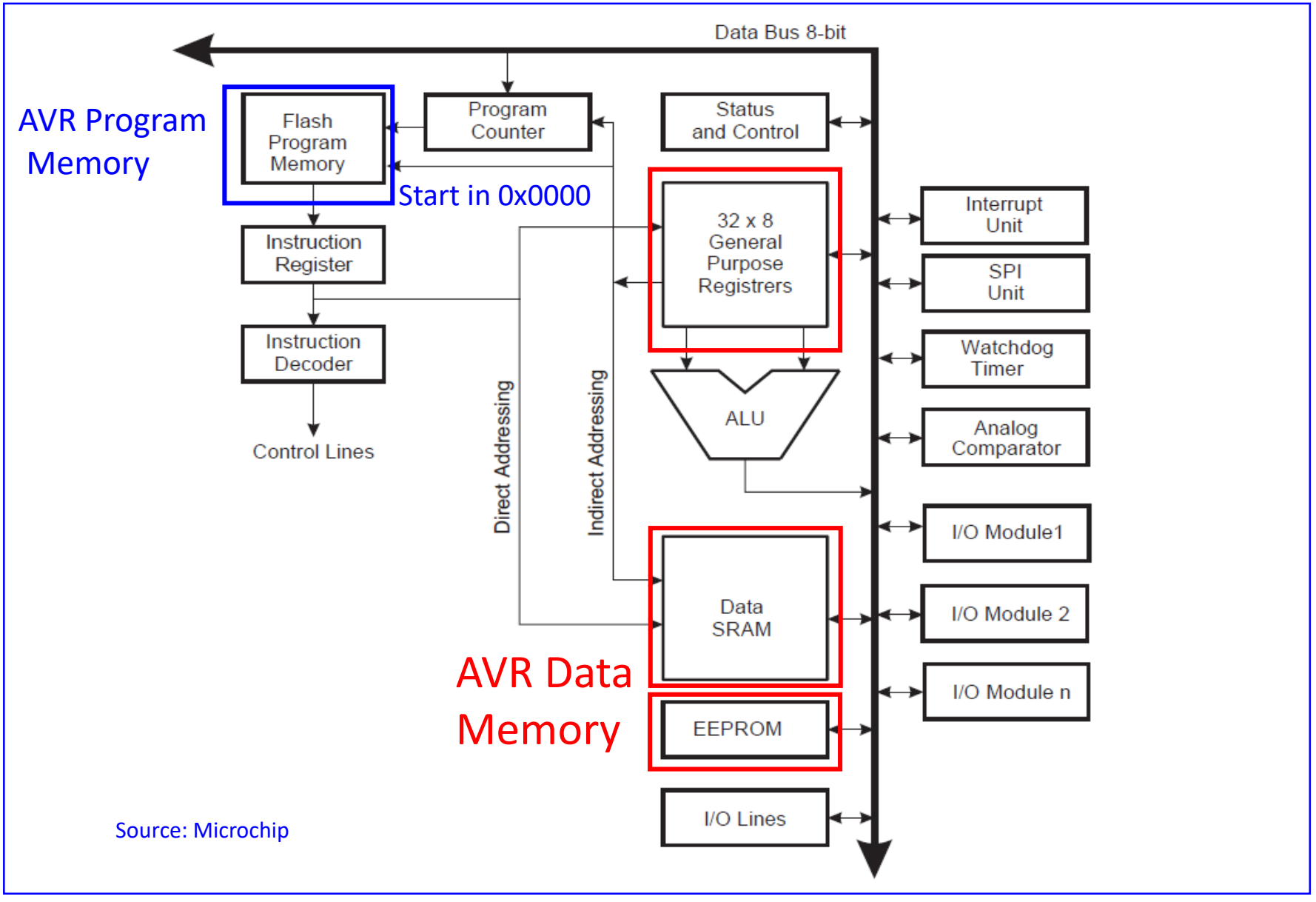
AVR Memory

AVR Memory: Harvard architecture



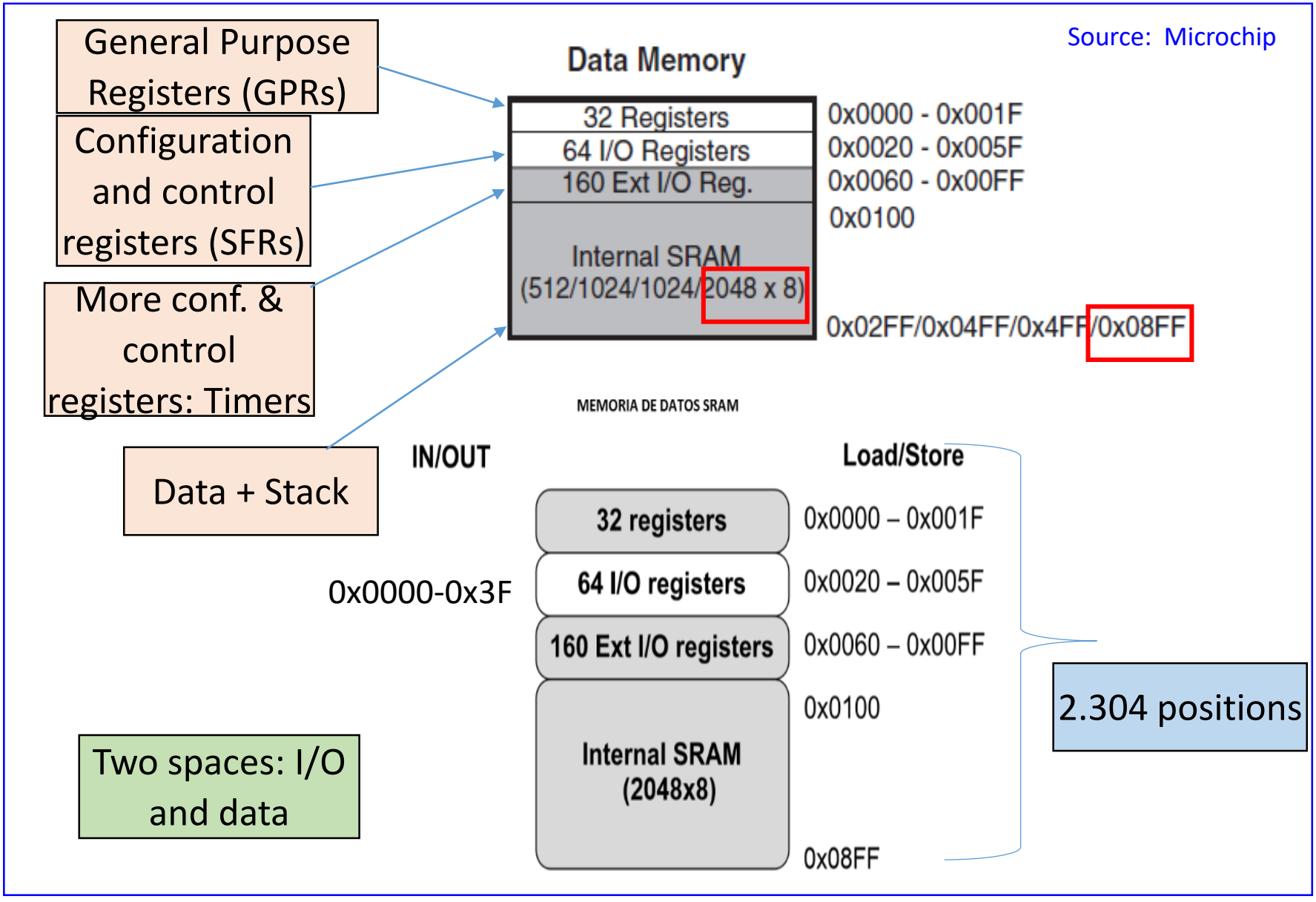
Source: "The AVR microcontroller and embedded system". M.Ali Mazidi, S.Naimi

AVR Architecture: Core



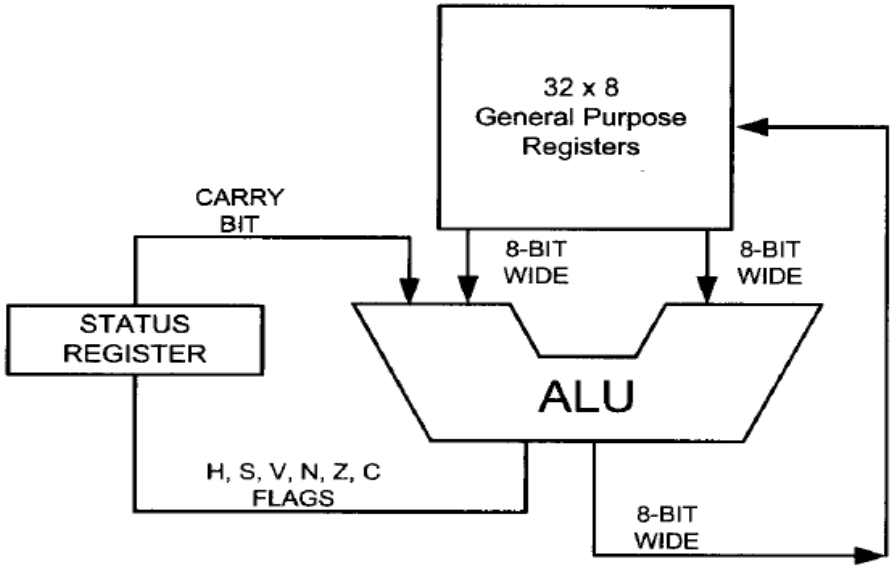
Source: Microchip

AVR Memory: SRAM Data Memory



AVR Memory: General purpose registers (GPRs)

- CPUs use registers to stores information temporarily.
- The information could be:
  - ✓ a byte of data to be processed
  - ✓ an address pointing to a data to be fetched.
- In AVR there are 32 general purpose registers, they are 8-bits wide. (R0-R31)
- The GPRs are located in the lowest location of memory address.
- The GPRs in AVR can be used by all the arithmetic and logic operations.



AVR Memory: EEPROM Data Memory

Source: Microchip

- 1Kbytes of data EEPROM memory
- Separate data space, single bytes can be read and written.
- 100,000 write/erase cycles
- Access I/O map
  - → EEPROM Data Register (EEDR)
  - → EEPROM Control Register (EECR)
  - → EEPROM Address (EEAR)

AVR Memory: EEPROM Data Memory

Write an AVR C program to store 'G' into location 0x005F of EEPROM.

Solution:

```
#include <avr/io.h>                 //standard AVR header
int main(void)
{
   while(EECR & (1<<EEWE));         //wait for last write to finish
   EEAR = 0x5f;                     //write 0x5F to address register
   EEDR = 'G';                      //write 'G' to data register
   EECR |= (1<<EEMWE);              //write one to EEMWE
   EECR |= (1<<EEWE);               //start EEPROM write
     return 0;
}
```

Source: "The AVR microcontroller and embedded system". M.Ali Mazidi, S.Naimi

AVR Memory: EEPROM Data Memory

Write an AVR C program to read the content of location 0x005F of EEPROM into PORTB.

Solution:

```
#include <avr/io.h>                 //standard AVR header
int main(void)
{
   DDRB = 0xFF;                     //make PORTB an output
   while(EECR & (1<<EEWE));         //wait for last write to finish
   EEAR = 0x5f;                     //write 0x5F to address register
   EECR |= (1<<EERE);               //start EEPROM read by writing EERE
   PORTB = EEDR;                    //move data from data register to PORTB
}
```

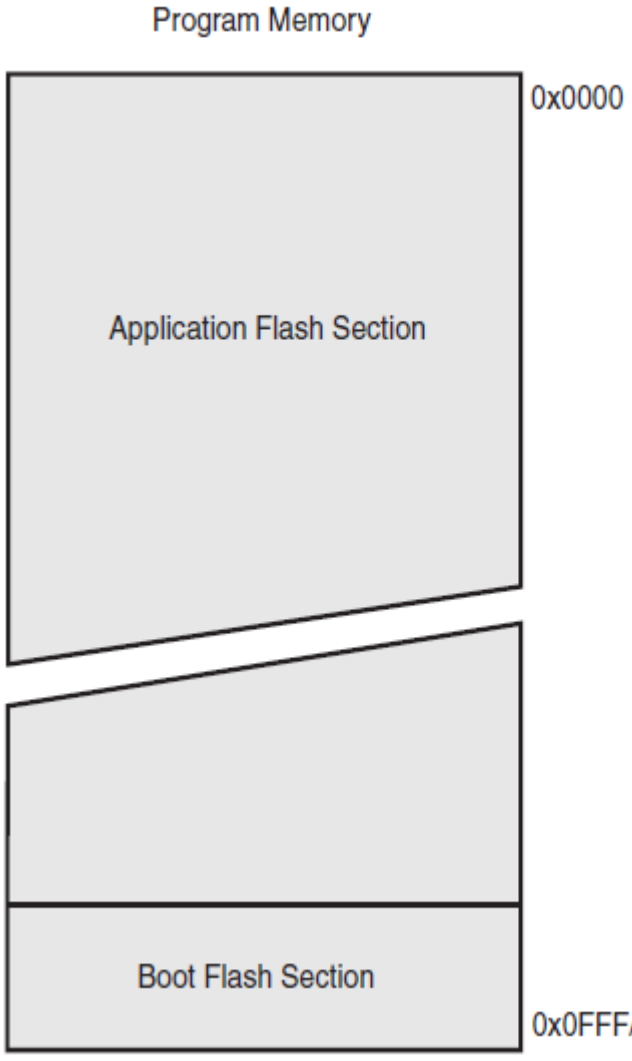
Source: "The AVR microcontroller and embedded system". M.Ali Mazidi, S.Naimi

AVR Memory: EEPROM Data Memory

```
#include <avr.io>
#include <avr/eeprom.h>

// Set 0x00 as EEPROM address to write
#define EEPROM_ADDRESS     (uint8_t *) 0x00 // (1-byte)

unsigned char myVar;

int main(){
  DDRD = 0xFF;                              // PORTD as output

  eeprom_write_byte(EEPROM_ADDRESS,'G');     // Write 'G'

  char c = eeprom_read_byte (EEPROM_ADDRESS);// Read eeprom
  PORTD = c;

  while (1);
}
```

Source: "The AVR microcontroller and embedded system". M.Ali Mazidi, S.Naimi

AVR Memory: In-System Reprogrammable Flash Program Memory



- Flash memory, 32KB , non-volatile.
- AVR instructions → 16 or 32 bits Flash is organized as 16K x 16.
- Little endian (High byte to high address)
- Program Counter (PC) → 14 bits addressing 16K program memory
- PC starts in 0x0000
- Instructions size: most of them 2 bytes, a few others 4 bytes
- Divided into two sections:
  - Boot Loader
  - Application Program
- 10.000 write/erase cycles.
- It is posible to store data also.

Source: Microchip