Lab 5 Memories and Index Registers

I. AVR ATmega2560 Memories

Program storage: Flash memory (.cseg directive)
Data storage: SRAM and EEPRROM (.dseg and .eseg directives)

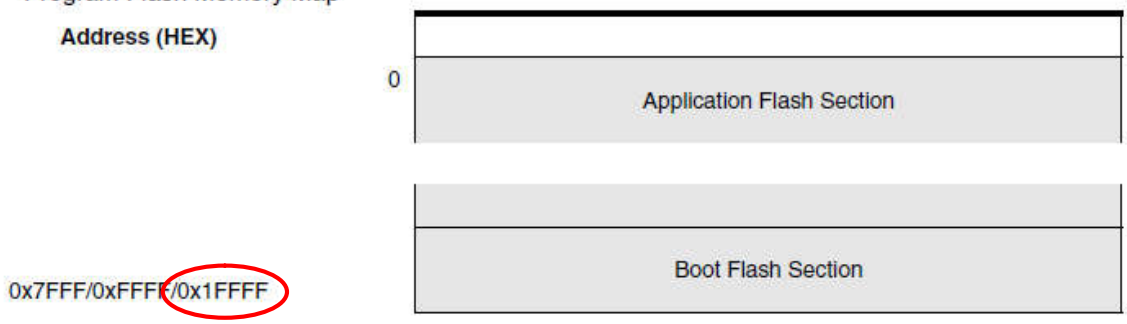
The Configuration Summary of ATmega2560:[1]

Table 2-1. Configuration Summary

Device	Flash	EEPROM	RAM	General Purpose I/O pins	16 bits resolution PWM channels	Serial USARTs	ADC Channels
ATmega2560	256KB	4KB	8KB	86	12	4	16

The diagram of the program flash memory[1]:

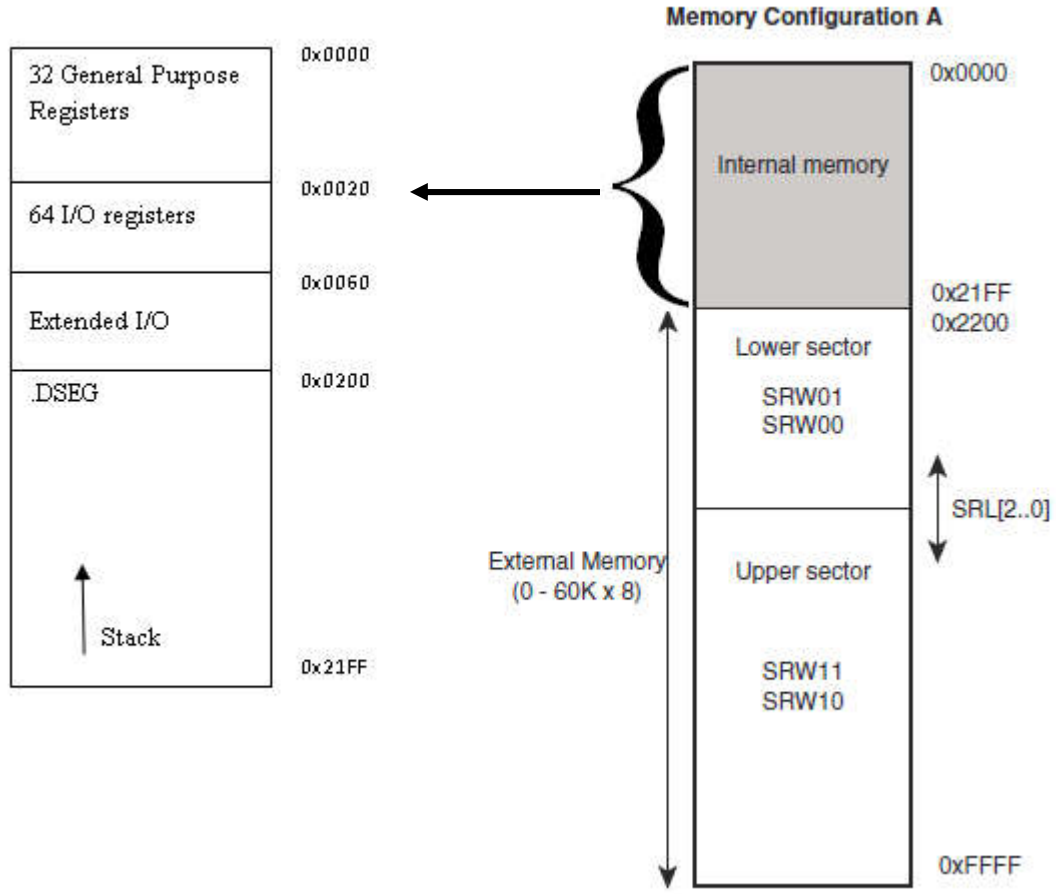
Figure 8-1. Program Flash Memory Map



What is the largest flash memory address of ATmega2560?
Based on Table 2-1, the memory size is 256KB -> 2^{18}Bytes ->2^{17}words, the largest word address is 0x1FFFF(17bits).

The diagram of the SRAM[1]:

Figure 9-1. External Memory with Sector Select



II. Index Registers

Data Direct Addressing:

We practiced the following instruction in the previous labs:

lds R0, msg
sts msg, R0

In general, the instruction takes the form of "lds Rd, (k)", where the value of k is a 16-bit unsigned integer representing memory address of the SRAM (data memory). The content (1 byte) stored in memory address k is loaded to register Rd.

Data Indirect Addressing:

The AVR processor has three register pairs that can be used for data indirect addressing. The three register pairs (also called index registers) are:

X -> R27:R26 or XH :XL
Y -> R29:R28 or YH :YL
Z -> R31:R30 or ZH :ZL

The address to be accessed must be preloaded into either X, Y, or Z register. What do the following statements do?

```
LD Rd, X
ST X, Rr
LPM R23, Z+
```

Indirect addressing is especially suited for accessing arrays, tables, and Stack Pointer.

III. Download lab5.asm and finish the program.

A C-style string (C string) is stored in the program memory (flash memory). Write a short program to calculate the length of the string and copy the string from the program memory (flash) to the data memory (SRAM).

IV. Download function1.asm and finish the program.

Without functions, lab5.asm has a problem. Two very different operations - copying a string, and measuring its length, are mixed together in the same code. Try finishing function1.asm such that each of these operations is confined to its own function. Arguments and expected return values are specified at the top of each function.

V. Download function2.asm and finish the program.

Without using the stack, function1.asm has undesirable *side effects* – values of registers in the calling program are changed as a side effect of running the program. Try using the stack to protect the values of the registers – except for the return value of str_len, the values of all the registers should be the same before and after the function call.