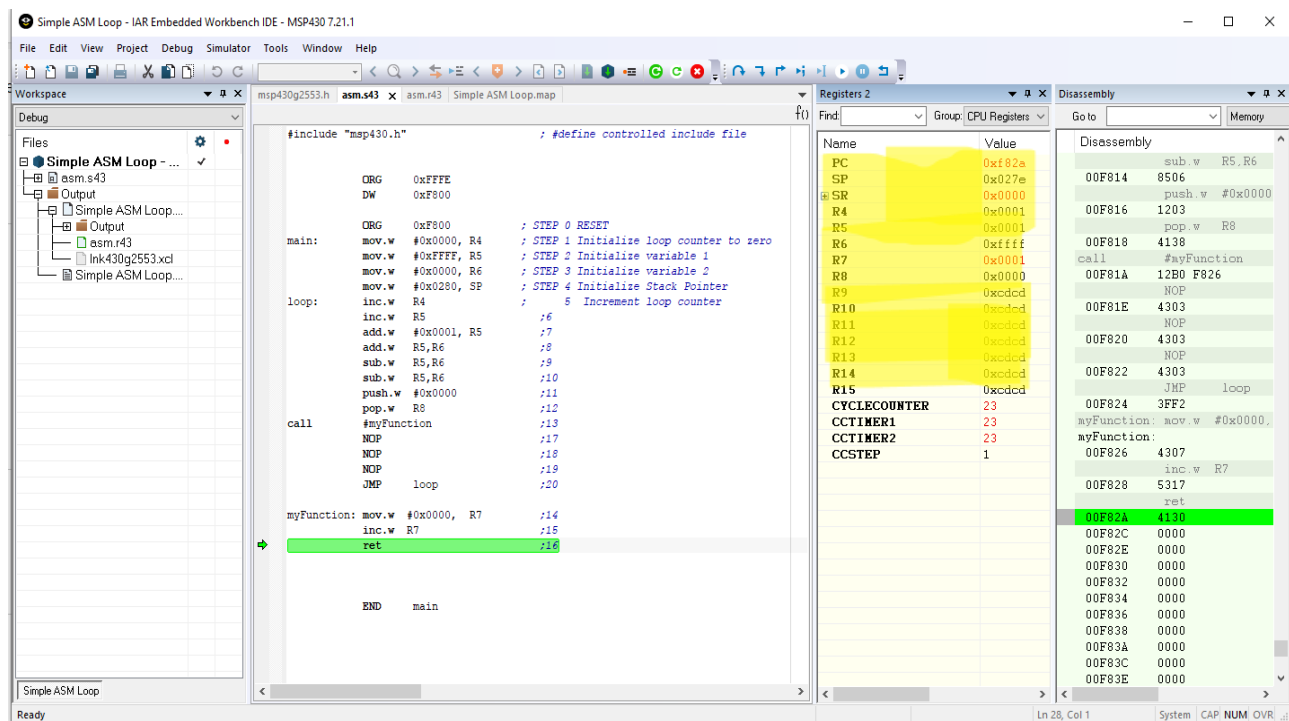


## LAB\_0 : IDE Familiarization Van Nguyen

### 1/- Registers - PC, SP, SR, and remaining registers :

Registers is a part of CPU, needed for the basic operation of the CPU, they are a set of 16 registers designated R0 – R15 needed to decode the instructions and implement them. The generous set of 16 registers is characteristic of a reduced instruction set computer (RISC).

- The first four registers ( R0 - R3 ) have dedicated functions with alternative names, such as the program counter ( PC/R0 ), stack pointer ( SP/R1 ), status register ( SR/CG1/R2 ), and constant generator ( CG2/R3 ).
- The remaining 12 registers ( R4 - R15 ) are general-purpose working registers. Words or bytes can be written to CPU registers, but the byte behavior is different from main memory: The destination is always the low byte, and the high byte is cleared (reset to 0).



View → Registers → Register 1



## 2/- Z, N, C, V flags :

The **Z, N, C, V flags** to be contained in the Status Register (SR) – The status register stores the state and control bits or flags.

**C** – Carry flag indicates that the last ALU operation produced a carry.

**Z** – Zero flag indicates that the last ALU operation resulted in 0.

**N** – Negative flag indicates that the last ALU operation resulted in a value  $< 0$ .

**V** – Overflow flag indicates that the last ALU operation on a signed variable overflowed the signed variables range.

The screenshot displays the IAR Embedded Workbench IDE for the MSP430 7.21.1. The interface is divided into several panes:

- Workspace:** Shows the project structure with files like `asm.s43`, `msp430f149.h`, and `IAB_0.map`.
- Debug:** A tree view showing the current debug session.
- Assembly Editor:** Displays the assembly code for `asm.s43`. The code includes a module definition, segment declarations, and an initialization routine for the stack. The line `init: MOV #SFE(CSTACK), SP` is highlighted.
- Registers:** A table showing the values of various registers. The Status Register (SR) is highlighted, showing bits for Reserved, V, SCG1, SCG0, OSGOff, CPDOff, GIE, N, Z, and C.
- Disassembly:** Shows the disassembled code, including the `init` routine and the `main` function. The instruction `MOV #SFE(CSTACK), SP` is highlighted.
- Memory:** A table showing the contents of memory locations, with addresses ranging from `0x0000` to `0x0020`.
- Debug Log:** A panel at the bottom for viewing debug messages.



### 3/- RAM : ( Random-Access Memory )

- RAM: SRAM, DRAM = volatile.
- Volatile: Loses its contents when power is removed
- Read or written with equal ease.

The screenshot displays the IAR Embedded Workbench IDE for the MSP430 7.21.1. The interface is divided into several panes:

- Workspace:** Shows the project structure with files like `asm.s43`, `msp430f149.h`, and `IAB_0.map`.
- Assembly Editor:** Displays the assembly code for `asm.s43`. The code includes headers, defines the module name, and sets up the stack. The current line is `init: MOV #SFE(CSTACK), SP ; set up stack`.
- Registers:** A table showing the values of CPU registers. The `SR` register is highlighted with a value of `0x0000`.
- Disassembly:** Shows the disassembled code, including the `init` section and the `main` function. The `init` section is highlighted, showing the `MOV #SFE(CSTACK), SP` instruction.
- Memory:** A table showing the memory layout. The `RAM` section is highlighted, showing the address range from `0x0200` to `0x03e0`.
- Debug Log:** A pane at the bottom for viewing debug messages.



#### 4/- FLASH :

- Nonvolatile memory : Retains its contents when power is removed and is therefore used for the program and constant data.
- Flash memory is the most common type of memory. It has largely superseded electrically erasable, programmable ROM (EEPROM)
- Flash memory must be erased before it can be written.

The screenshot displays the IAR Embedded Workbench IDE interface for the MSP430 7.21.1. The main window shows the assembly code for a program named 'main'. The code includes a pre-declaration of the stack segment and an initialization routine to set up the stack pointer.

```

#include "msp430.h"          ;define controlled include file

NAME main                  ; module name

PUBLIC main                ; make the main label visible
                           ; outside this module

ORG 0FFFFh                 ; set reset vector to 'init' label
DC16 init

RSEG CSTACK                ; pre-declaration of segment
RSEG CODE                  ; place program in 'CODE' segment

init: MOV #SFE(CSTACK), SP ; set up stack

```

The Registers window shows the CPU registers, with the PC register at 0x1100 and the SP register at 0x0000. The Disassembly window shows the assembly code being executed, including the initialization routine and the main function.

The Memory window shows the memory layout, with the FLASH memory starting at 0x1100 and the main function starting at 0x1104.

