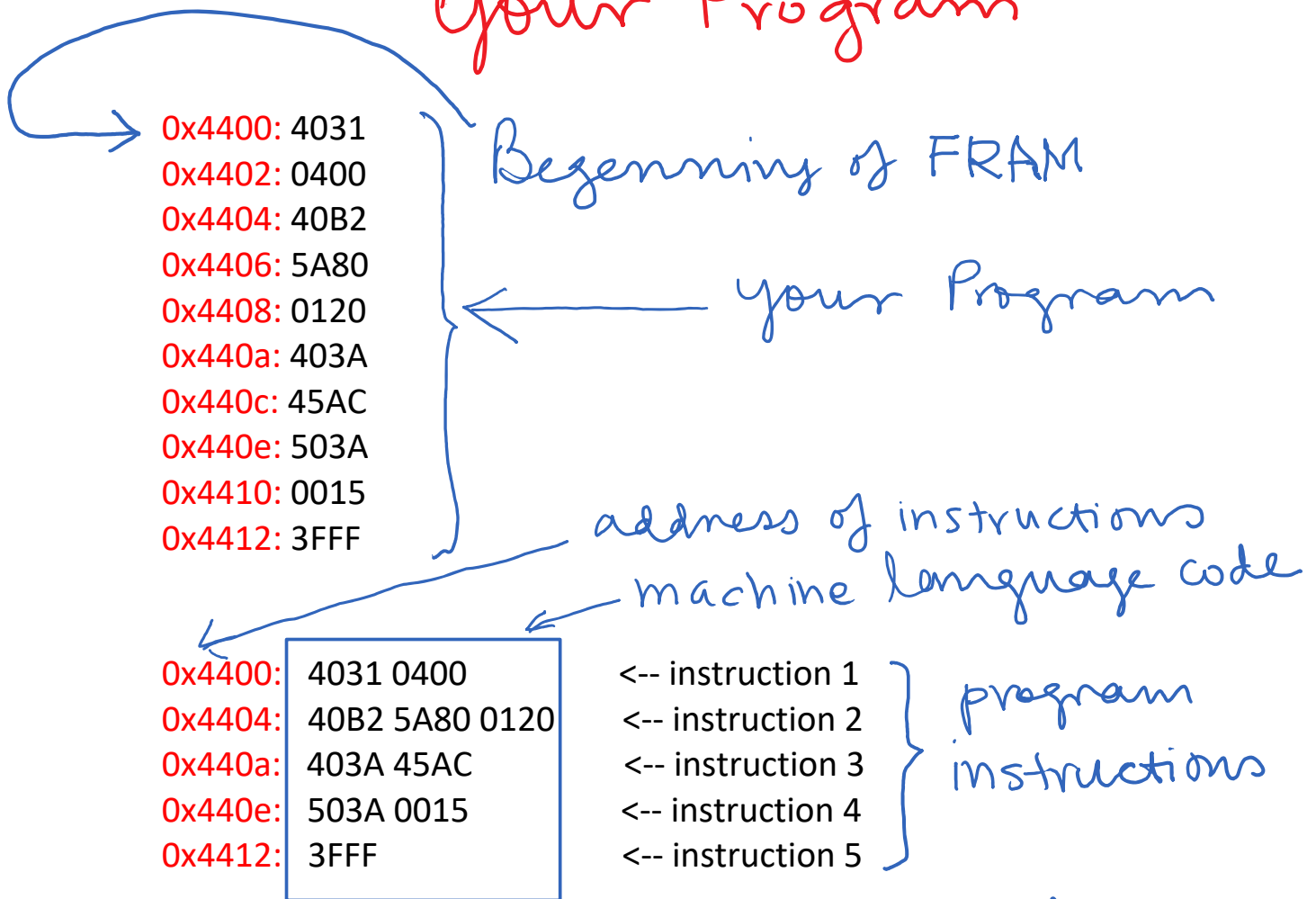


Your Program



Each instruction has an address

4400: 4031 0400
4404: 40B2 5A80 0120
440a: 403A 45AC
440e: 503A 0015
4412: 3FFF

```
MOV.W #0x0400, SP
MOV.W #0x5a80,&Watchdog_Timer_WDTCTL
MOV.W #0x45ac, R10
ADD.W #0x0015, R10
loop:
JMP     loop
```

Assembly Language

Assembly Language is easier to understand for humans (english language like)

```
MOV.W #0x0400, SP
MOV.W #0x5a80, &Watchdog_Timer_WDTCTL

MOV.W #0x45ac, R10
ADD.W #0x0015, R10

loop: JMP    loop
```

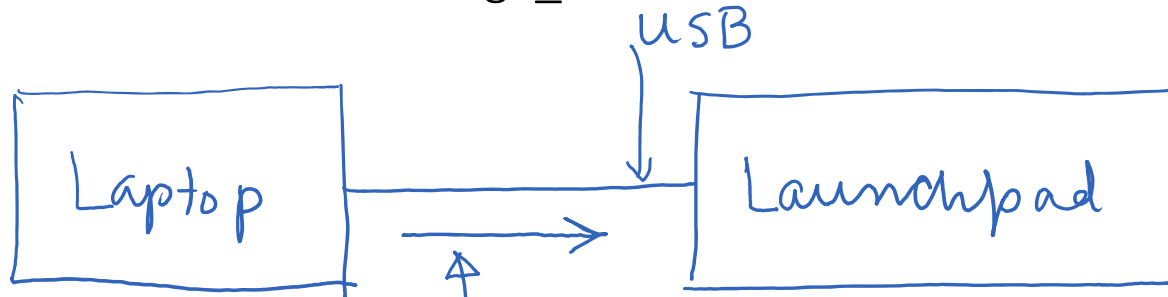
you input
assembly
language in
the editor of
your IDE
Code Composer
Studio

IDE :
Integrated
Development
Environment

you compile (build)
your program in your
IDE

```
4031 0400
40B2 5A80 0120
403A 45AC
503A 0015
3FFF
```

compilation converts
assembly
language
instructions
into machine
language code

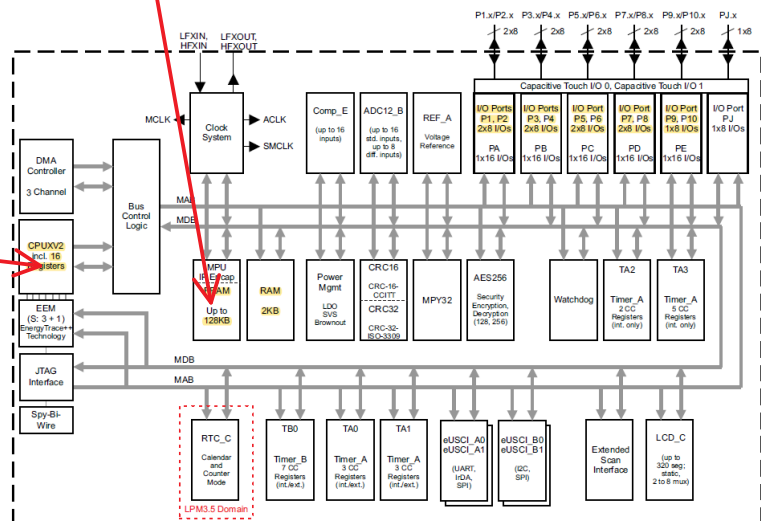


upload machine code to the MCU
(programming the MCU)

machine code moves from the
IDE to FRAM in the MCU

0x4400: 4031 0400
0x4404: 40B2 5A80 0120
0x440a: 403A 45AC
0x440e: 503A 0015
0x4412: 3FFF

CPU executes
machine code
instructions
one by
one



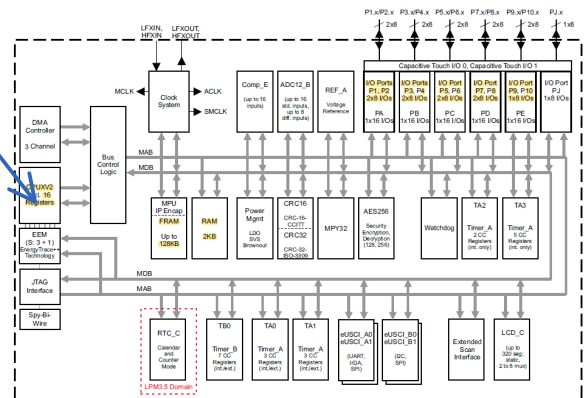
16 Core Registers
(16 Bit Wide)

Program Counter	PC/R0
Stack Pointer	SP/R1
Status Register	SR/CG1/R2
Constant Generator	CG2/R3
General-Purpose Register	R4
General-Purpose Register	R5
General-Purpose Register	R6
General-Purpose Register	R7
General-Purpose Register	R8
General-Purpose Register	R9
General-Purpose Register	R10
General-Purpose Register	R11
General-Purpose Register	R12
General-Purpose Register	R13
General-Purpose Register	R14
General-Purpose Register	R15

Like in the CPU
Are not memory mapped

They are known
by their names

$R_0, R_1, \dots, R_{15}$



Program Counter

PC/R0

PC contains the address of the next instruction to be executed by the CPU

Download Data Sheets



LaunchPad's User's Guide (SLAU627A)

MSP430FR698x Specific Data Sheet (SLAS789D)

MSP430FR6xx Family User's Guide (SLAU3670)

Google SLAU627A, SLAS789D and SLAU3670 to locate the pdf
Data sheets

Download Code Composer Studio
(CCS) V9.1.0

you will use CCS to

Write and edit your assembly language programs

Compile (build) your assembly language program into machine language code

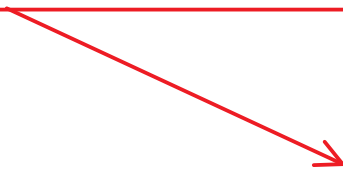
Load the program into FRAM of the MCU

Debug your program as it runs in the CPU.

Single Step your program running in the CPU

Set Break Points

Each instruction takes a certain number of clock cycles to execute.



MOV.W #0x0400, SP	
MOV.W #0x5a80, &Watchdog_Timer_WDTCTL	5 Clock Cycles
MOV.W #0x45ac, R10	2 Clock Cycles
ADD.W #0x0015, R10	2 Clock Cycles
loop: JMP loop	2 Clock Cycles

Using default settings the clock of your MCU is running at $\sim 1\text{MHz}$
i.e one clock cycle is $1\mu\text{sec}$ long.

$\therefore$ MOV.W #0x45ac, R10 takes $2\mu\text{secs}$ to execute

CCS v5 will let you figure out how many clock cycles are used up by each instruction