your program will sit in Flash (ROM)

Central Processing Unit (CPU) is the brain of the MCU

Variables will sit in RAM

Ports are for Digital GPIO (General Purpose Input - Output)
Pin 2 can be connected to:

- \( V_{CC} \) (Supply Voltage)
- Ground

Port 1:
- GP10 (Port 1 Pin 0)
- Timer A0 Clock (TA0CLK)
- Auxiliary Clock (ACLK)

Peripheral:
- Port P1
- Timer0-A3
- Clock System
- etc.
Your Assembly Language program is typically stored in FLASH (ROM) memory. The instructions of the program are fetched from ROM, one by one, into the CPU and executed there.
Data Transfer using a tri-state buffer

Registers have addresses "Memory Mapping"

Registers

EF = 00 → Reg A's output connected to Bus others put Z on the Bus
EF = 01 → Reg B's
EF = 10 → Reg C's
EF = 11 → Reg D's

2-bit "Address Bus"

8-bit "Data Bus"
Memory Mapped

Every peripheral register is mapped to an address, including the memory registers of FLASH (ROM) and RAM.

Address Bus $\rightarrow 16$ bit wide

addresses registers

(HEX)

$\leftarrow 16$ bit $\rightarrow 8$ bit $\uparrow$

$0000\, 0001$ ... $FFFF$ total number of registers that can be addressed $= 65536 = 2^{16}$

$0000\, 0000$ ... $FFFF$

8 bit peripheral registers

16 bit peripheral registers

RAM memory registers

FLASH $\rightarrow$ ROM memory registers
von Neumann Architecture  
(Princeton Architecture)

MSP430 Architecture

one set of addresses for ROM and RAM + other peripheral registers

Harvard Architecture

Separate set of addresses for ROM

Separate set of addresses for RAM
Harvard Architecture

Efficient:
Simultaneous access to the program and data memories

Complex:
Constants (in ROM) and variables (in RAM) live in different address spaces and must be treated differently

von Neumann Architecture
(Princeton Architecture)

Not as efficient as Harvard Architecture

Simpler:
Constants (in RAM) and variables (in RAM) live in the same address space and are treated similarly