Microcomputers I – CE 320

Mohammad Ghamari, Ph.D.
Electrical and Computer Engineering
Kettering University

Lecture 3:
Introduction to HCS12/9S12

Announcement

- You are going to have your first quiz on Thursday, October 19.
  - Covers Intro to Microcomputers
  - Number Systems

Today’s Topics

- HSC12 architecture details
- Major pieces of information about the processor
- Specific information on Freescale HCS12 microcontroller

HCS12 Architecture Details

The microcontrollers in the 9S12 family differ by the amount of memory and by the types of I/O modules

- All 9S12 microcontrollers have:
  - a 16-bit central processing unit (**HCS12CPU**),
  - a system integration module (SIM),
  - RAM (volatile random access memory),
  - Flash EEPROM (nonvolatile electrically erasable programmable read only memory),
  - a phase-locked loop (PLL).
- The 9S12 microcontrollers are configured with zero, one, or more of the following modules:
  - asynchronous serial communications interface (SCI),
  - serial peripheral interface (SPI),
  - inter-integrated circuit (I2C),
  - Key wakeup,
  - 16-bit timer,
  - a pulse width modulation (PWM),
  - 10-bit or 12-bit analog-to-digital converter (ADC),
  - 8-bit digital-to-analog converter (DAC),
  - liquid crystal display driver (LCD),
  - controller area network (CAN 2.0),
  - universal serial bus (USB 2.0) interface,
  - Ethernet (MAC FEC 10/100) interface,
  - memory expansion logic.

HCS12 Architecture Details

The microcontrollers in the 9S12 family differ by the amount of memory and by the types of I/O modules

- All 9S12 microcontrollers have:
  - a 16-bit central processing unit (**HCS12CPU**),
  - a system integration module (SIM),
  - RAM (volatile random access memory),
  - Flash EEPROM (nonvolatile electrically erasable programmable read only memory),
  - a phase-locked loop (PLL).
- The 9S12 microcontrollers are configured with zero, one, or more of the following modules:
  - asynchronous serial communications interface (SCI),
  - serial peripheral interface (SPI),
  - inter-integrated circuit (I2C),
  - Key wakeup,
  - 16-bit timer,
  - a pulse width modulation (PWM),
  - 10-bit or 12-bit analog-to-digital converter (ADC),
  - 8-bit digital-to-analog converter (DAC),
  - liquid crystal display driver (LCD),
  - controller area network (CAN 2.0),
  - universal serial bus (USB 2.0) interface,
  - Ethernet (MAC FEC 10/100) interface,
  - memory expansion logic.

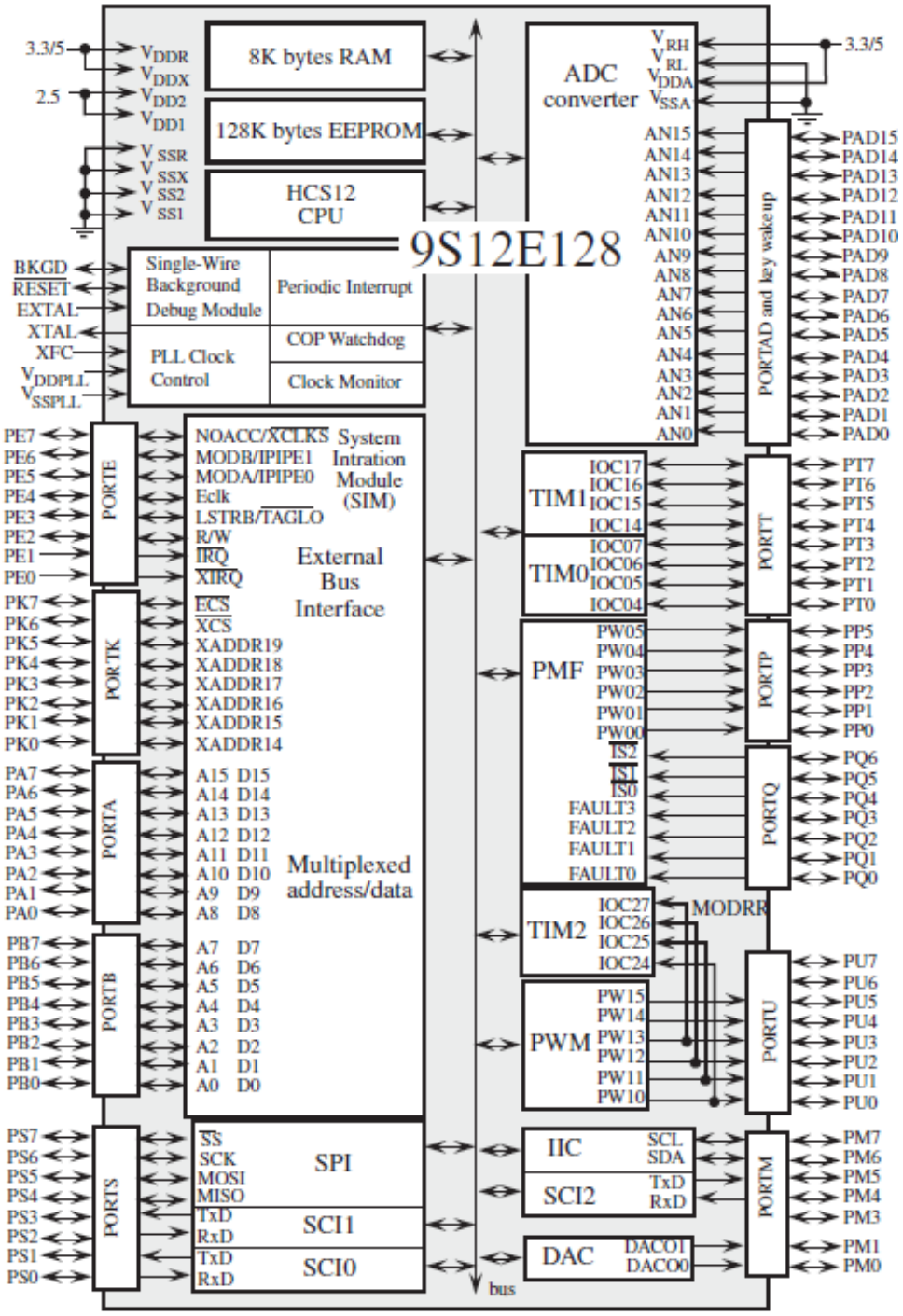
Remember

Semiconductor Memory Types:

RAM: Random Access Memory (can read and write)
ROM: Read Only Memory (programmed at factory)
PROM: Programmable Read Only Memory
(Programmed once at site)
EPROM: Erasable Programmable Read Only Memory
(Program at site, can erase using UV light and reprogram)
EEPROM: Electrically Erasable Programmable Read Only Memory
(Program and erase using voltage rather than UV light)

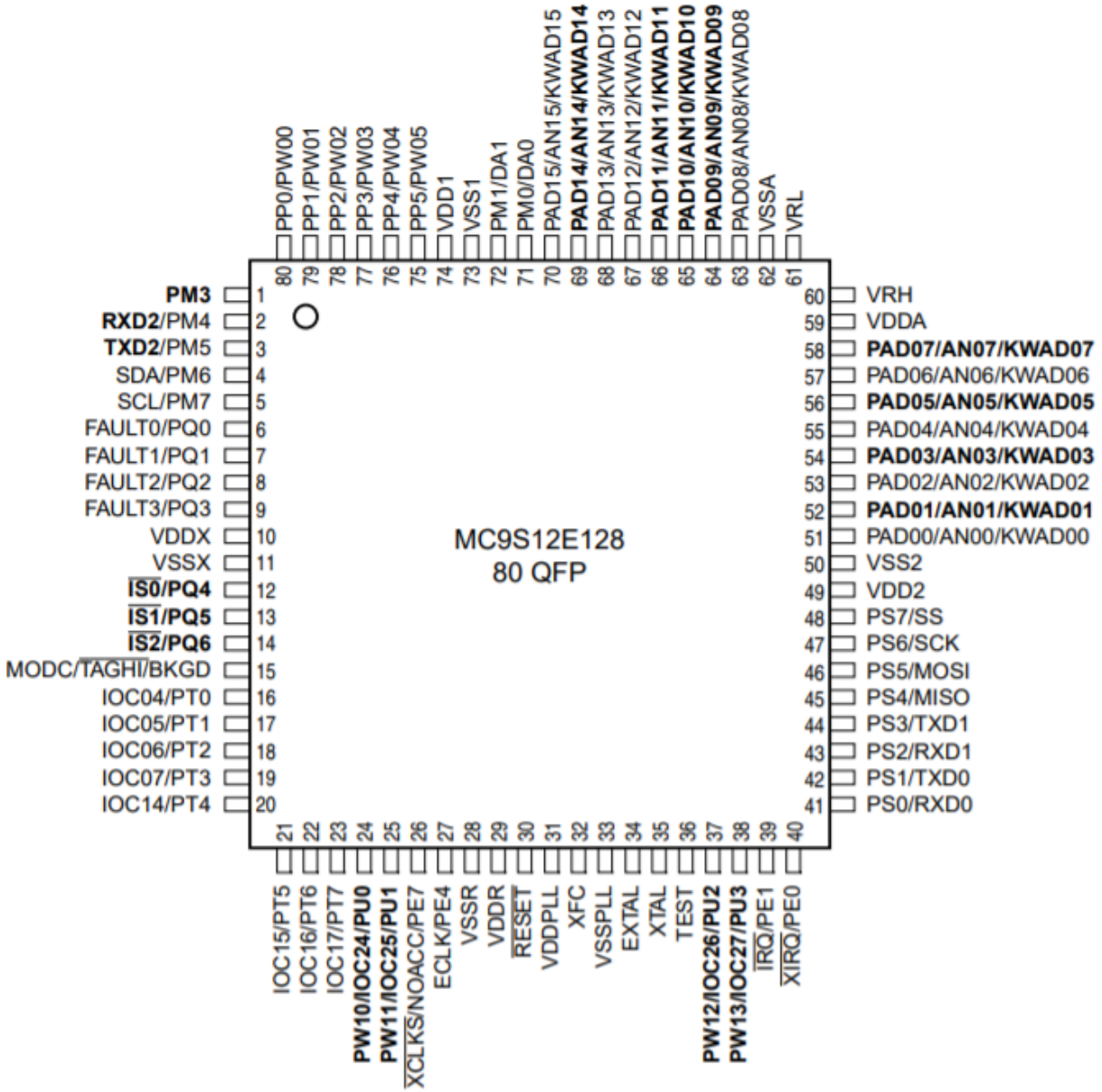
HCS12 Architecture Details. 9S12E128

- 8 KB of RAM, 128 KB of flash EEPROM
- 12 input capture/output compare timer pins,
- 12 pulse-width modulated output pins,
- 16 ADC inputs, two DAC outputs,
- one SPI modules, three SCI modules, one I2C interface.
- There are two sizes of the 9S12E128 chip one with 80 pins and the other with 112 pins.
- The 112-pin chip has 92 I/O pins
- We clear (0) a bit in the direction register to make that pin an input and set it (1) to make it an output



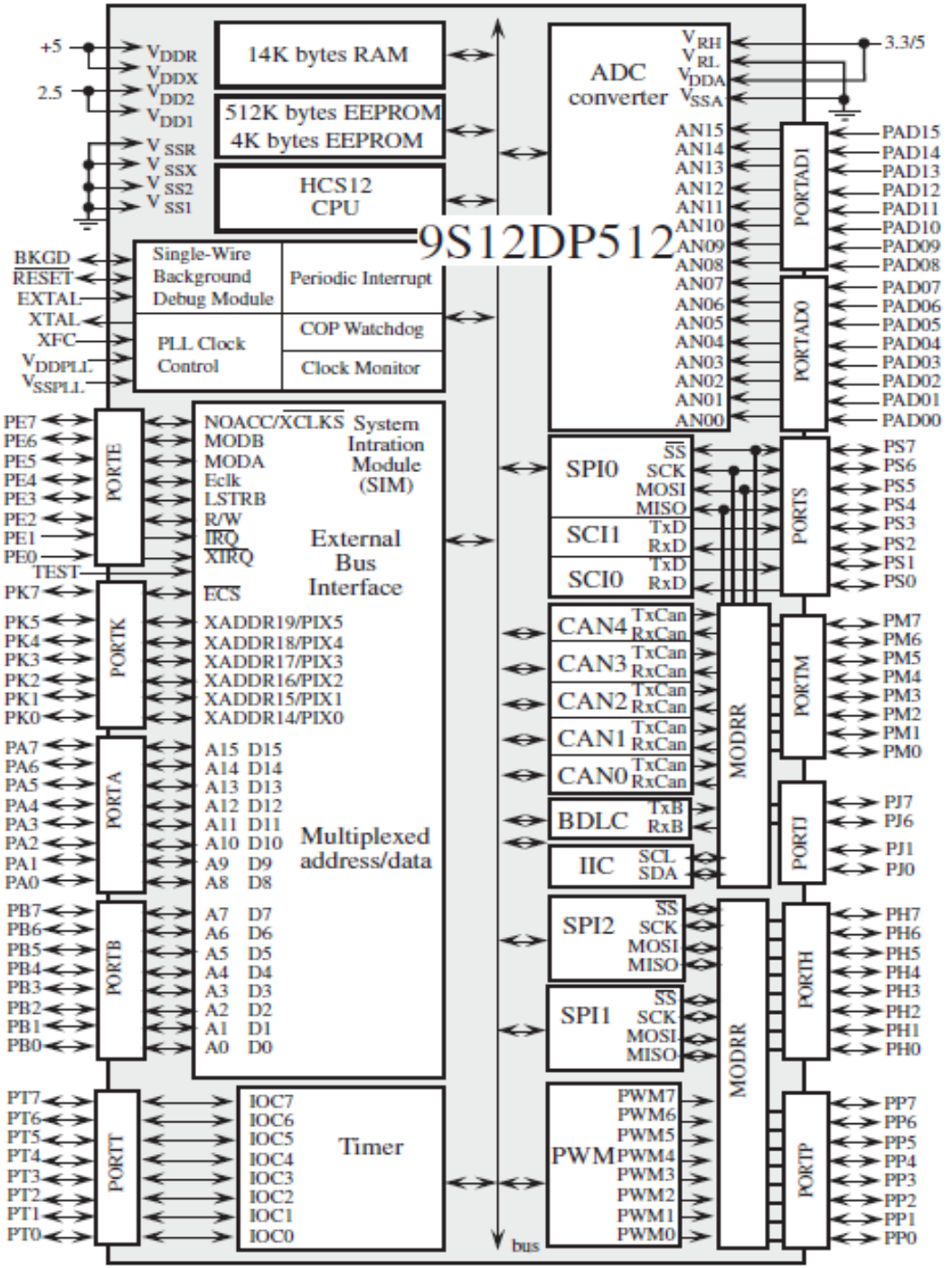
HCS12 Architecture Details.

MC9S12E128 Pin Assignments



HCS12 Architecture Details. 9S12E512

- **512** KB EEPROM
- 91 I/O pins
- Etc. Etc.
- We clear (0) a bit in the direction register to make that pin an input and set it (1) to make it an output

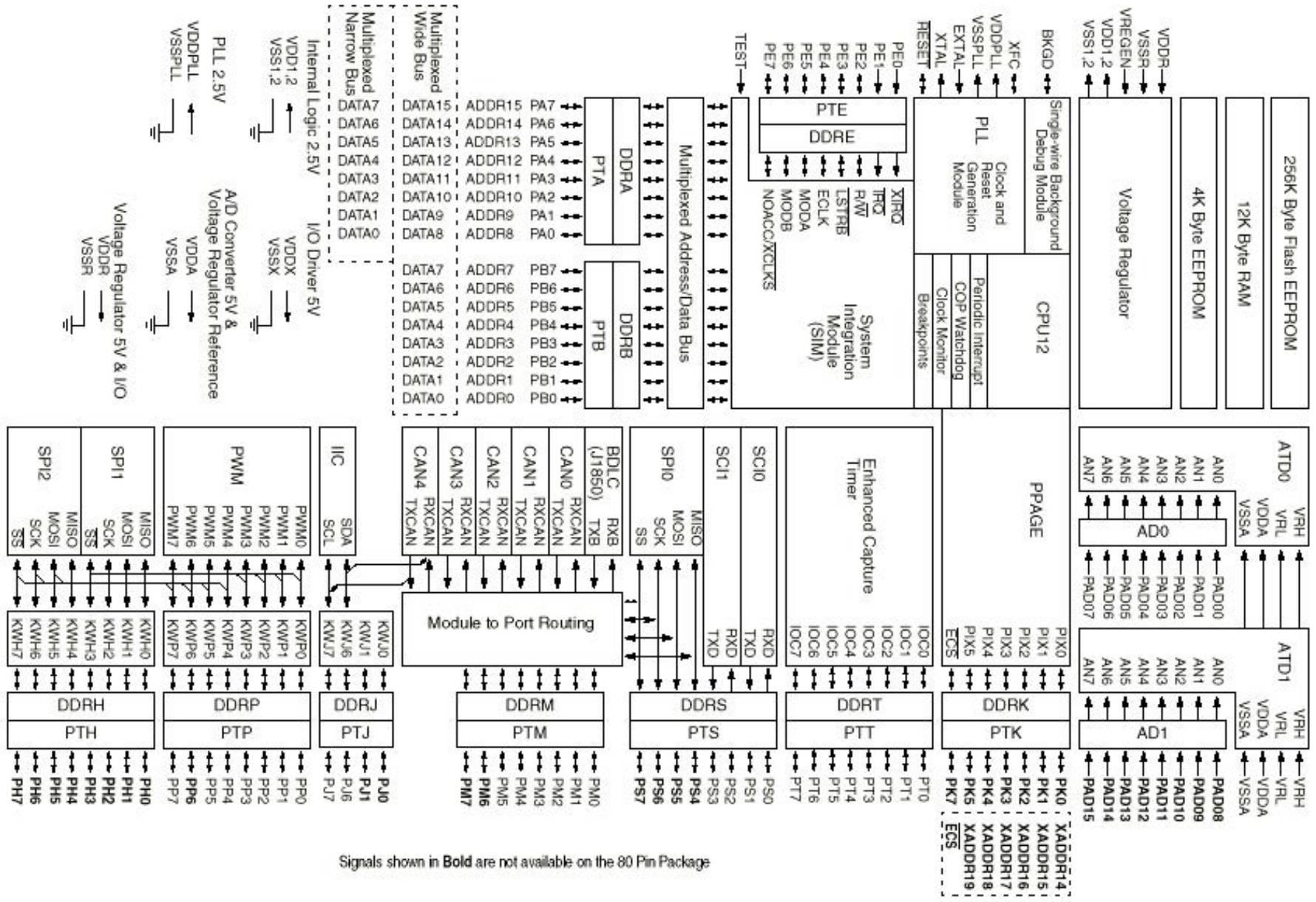


HCS12 Architecture Details. MC9S12DG256

- 16-bit CPU (central processing unit), 256KB of flash memory, 12KB of RAM, 4KB of EEPROM and many on-chip peripherals
- The main features of the MC9S12DG256 are listed below:
  - Powerful 16-bit CPU
  - 256K bytes of flash memory
  - 12K bytes of RAM
  - 4K bytes of EEPROM
  - 2 SCI ports
  - 3 SPI ports
  - 2 CAN 2.0 ports
  - I^2C interface
  - 8-ch 16-bit timers
  - 8-ch 8-bit or 4-ch 16 bit PWM
  - 16-channel 10-bit A/D converter
  - Fast 25 MHz bus speed via on-chip Phase Lock Loop
  - BDM for in-circuit programming and debugging
  - 112-pin LQFP package offers up to 91 I/O in a small footprint

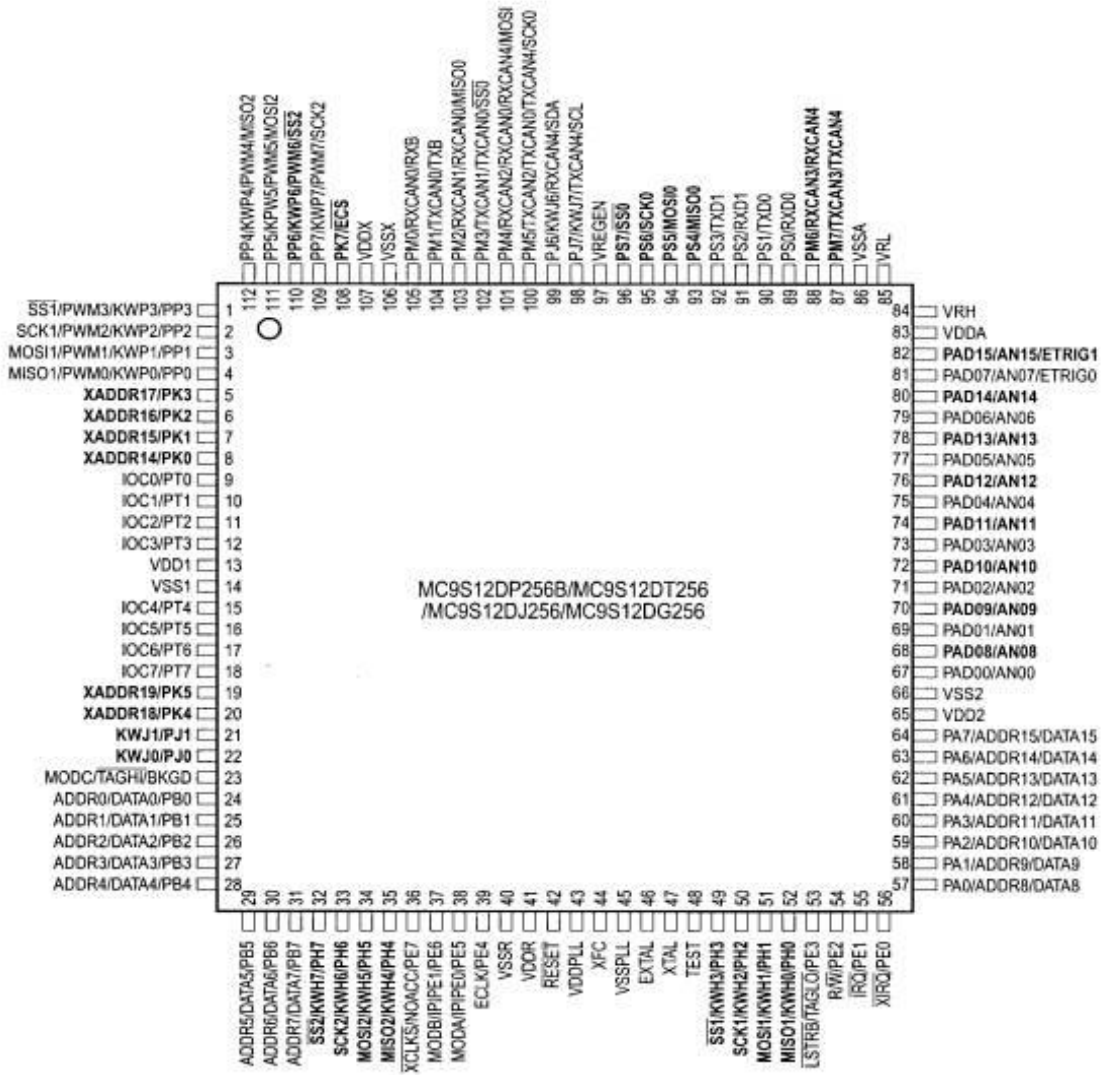
HCS12 Architecture Details.

MC9S12DG256 Block Diagram



HCS12 Architecture Details.

MC9S12DG256 Pin Assignments



Signals shown in **Bold** are not available on the 80 Pin Package

HCS12 Registers

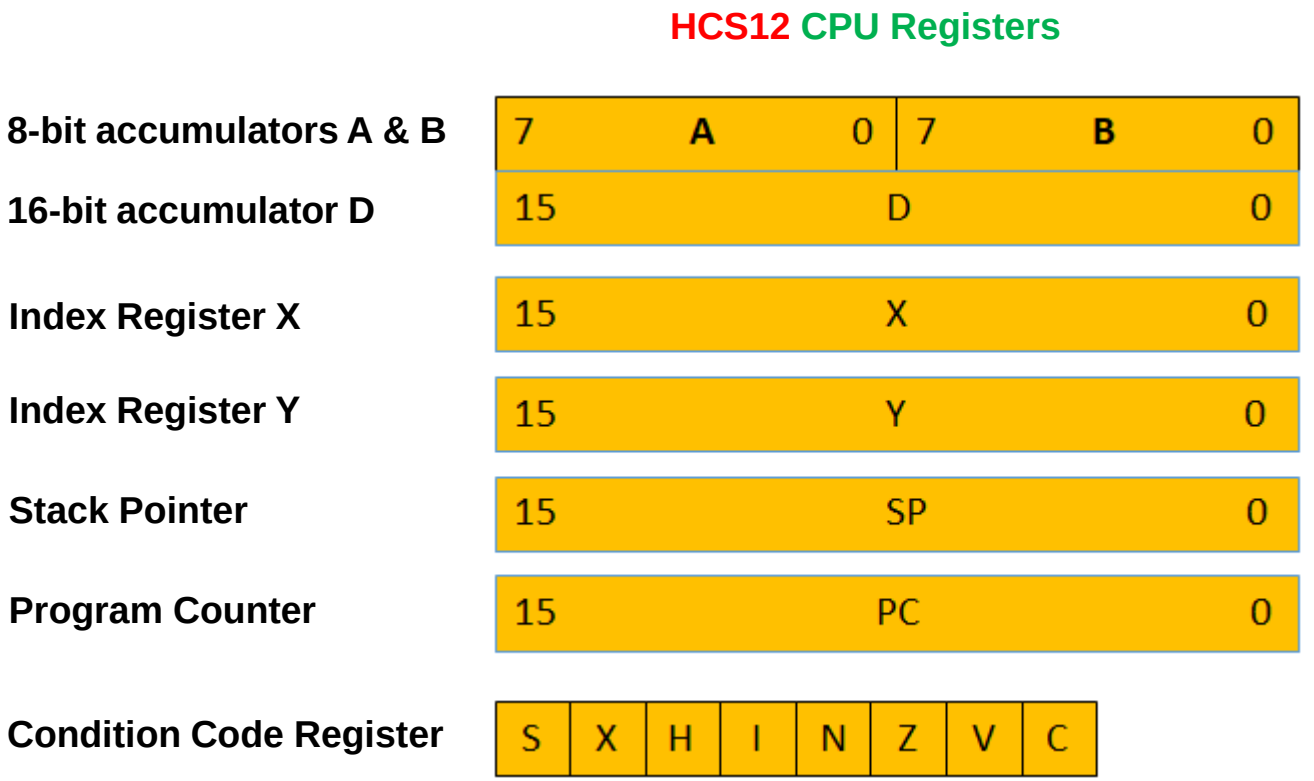
Remember that a register is a storage location (groups of D flip-flops) inside the MCU.

- The HCS12 MCU has two types of registers:
  - CPU registers
    - Mainly used to perform general-purpose operations such as arithmetic, logic, and program flow control.
    - Do not occupy the HCS12 memory space.
  - I/O registers
    - Mainly used to configure the operations of peripheral functions &
    - To hold data transferred in and out of peripheral subsystem &
    - To record the status of I/O operations.
    - Are treated as memory locations when they are accessed.
- The I/O registers are further classified into:
  - **Data** register
  - **Data direction** register
  - **Control** register
  - **Status** register

You will learn about I/O registers later.

Register Set (Programming Model)

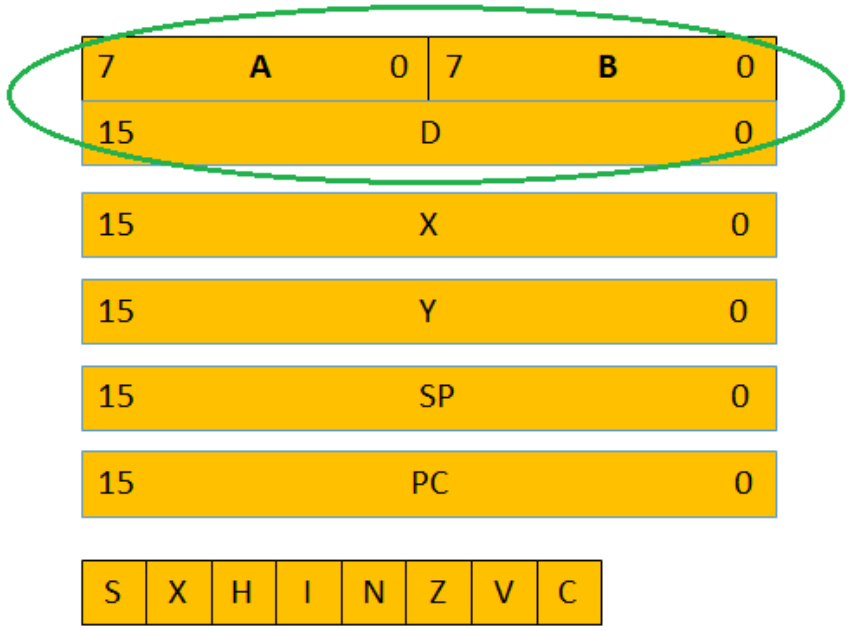
- The register set is also called the **programming model** of the computer.
- Programming Model
  - An abstract model of the microprocessor registers
  - This provides enough detail to understand the fundamentals of programming.
- In many processors, data may only be operated on if it is in a register.



CPU Registers

General Purpose Registers

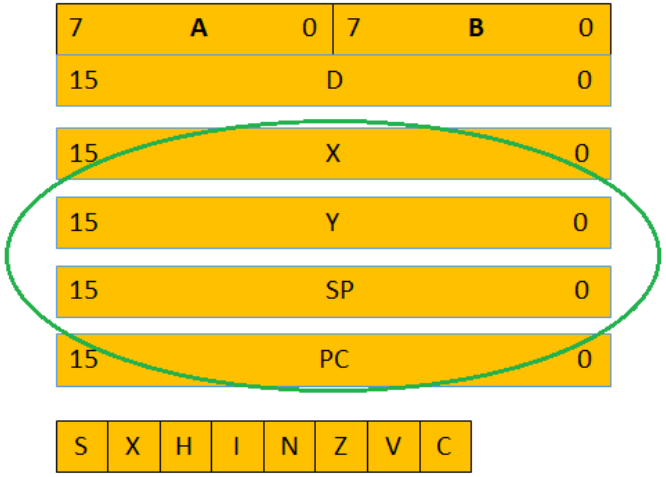
- A
  - A one-byte (8-bit) general purpose register.
  - Since many mathematical operations can be performed using A, it is also referred to as the **A accumulator**.
- B
  - A one-byte (8-bit) general purpose register.
  - Since many mathematical operations can be performed using B, it is also referred to as the **B accumulator**.
- D
  - A two-byte (16-bit) general purpose register.
  - The D register is actually the concatenation of the A and B registers.
  - A is used as the more significant byte with B as the less significant byte.
  - **Note**: The two bytes worth of registers may be used as either A and B or as D, but not both at the same time.



CPU Registers

Index Registers and Others

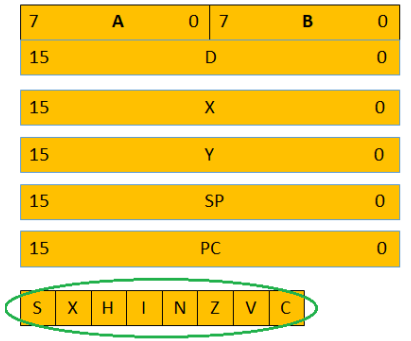
- X
  - A two-byte (16-bit) register primarily used to hold addresses. Very few mathematical operations can be performed.
- Y
  - A two-byte (16-bit) register primarily used to hold addresses. Very few mathematical operations can be performed.
- SP
  - A two-byte (16-bit) register used to manipulate the stack data structure.
- PC
  - Called the **program counter**, this is a two-byte (16-bit) register that holds the address of the **<u>next</u>** instruction to be executed.

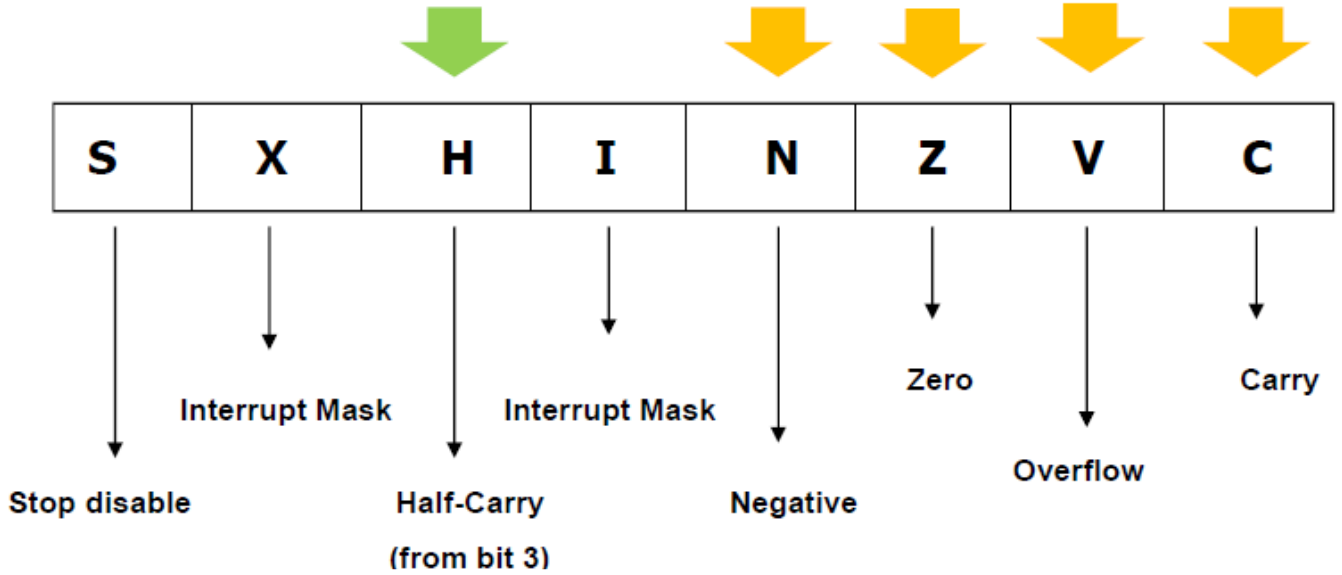


CPU Registers

Condition Code Register

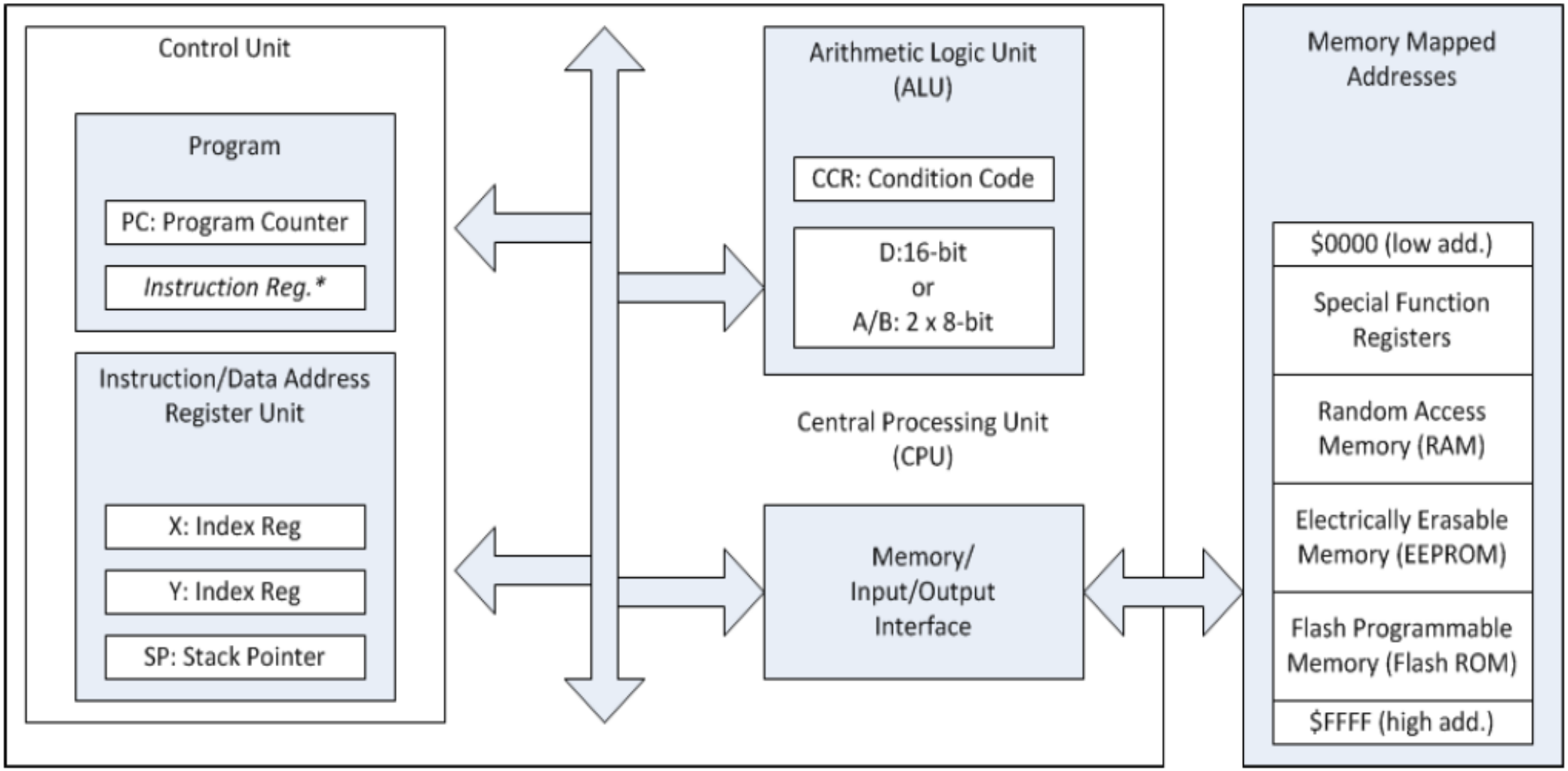
- CCR is a 8-bit register
- Used to keep track of the program execution status
- Control the execution of conditional instructions
- Enable the interrupt handling
- Most important for arithmetic (5 bits for this); 3 bits for control





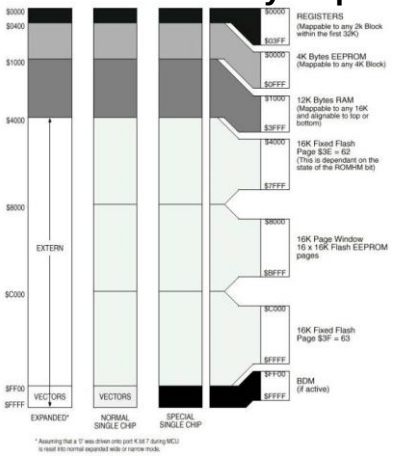
CPU Registers

Registers Interaction – simple processor diagram



Memory Model

MC9S12DG256 Memory Map

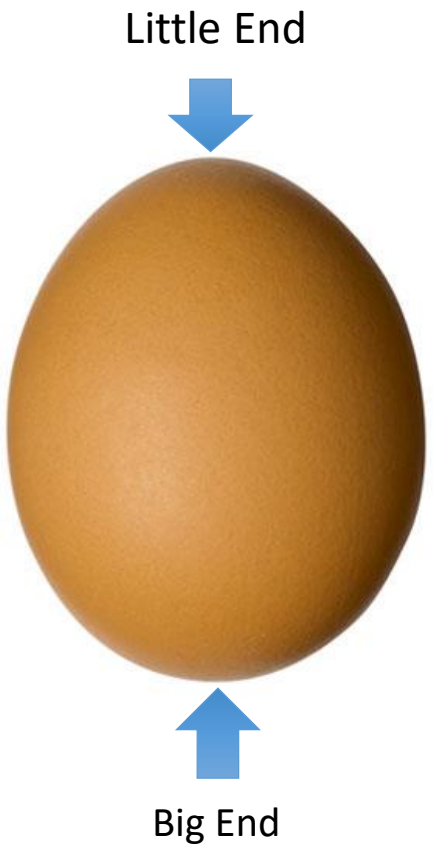


- **Memory model** includes details on how the microcomputer stores information internally.
- Programmers usually visualize memory as a bunch of sequential spaces.
- Each space has a unique address that is used to refer the location.
- Number of memory units
  - Remember the two different architectures: Princeton* and Harvard
- Bit size of each location
  - The number of bits stored in each location
- Bit size of the address
  - The number of bits used for the address limits the number of memory location

0000	B6
	.
	.
	.
A128	B6
A129	C1
A12A	12
	.
	.
	.
FFFE	32
FFFF	73

Endianness

Gulliver's Travels



Big-endians crack soft-boiled eggs at the big end, and little-endians crack them at the other end in the story.

Endianness

Big and Little-endian

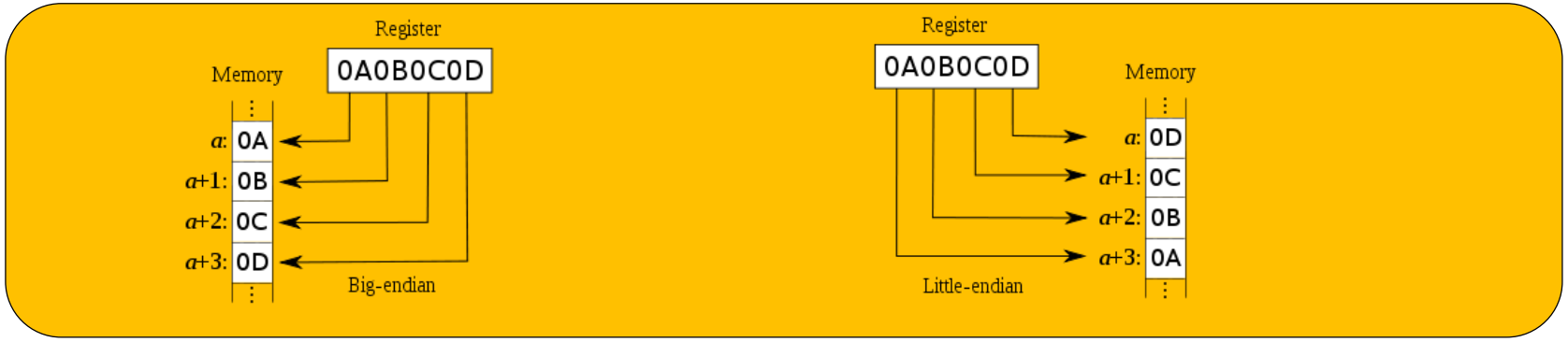
- A microprocessor may need to store a number that is larger than a single memory location (in the HCS12, the size of memory location is 1 byte).

- **Q:** How to store 16-, 32- or 64-bit word to 8-bit address space?

- Endianness means which byte is put **first** into the memory!

  - **Big**-endian (**HCS12**): put the big number portion of the large number **first** into the memory (the Most Significant Byte (MSB) occupies the lowest address space).

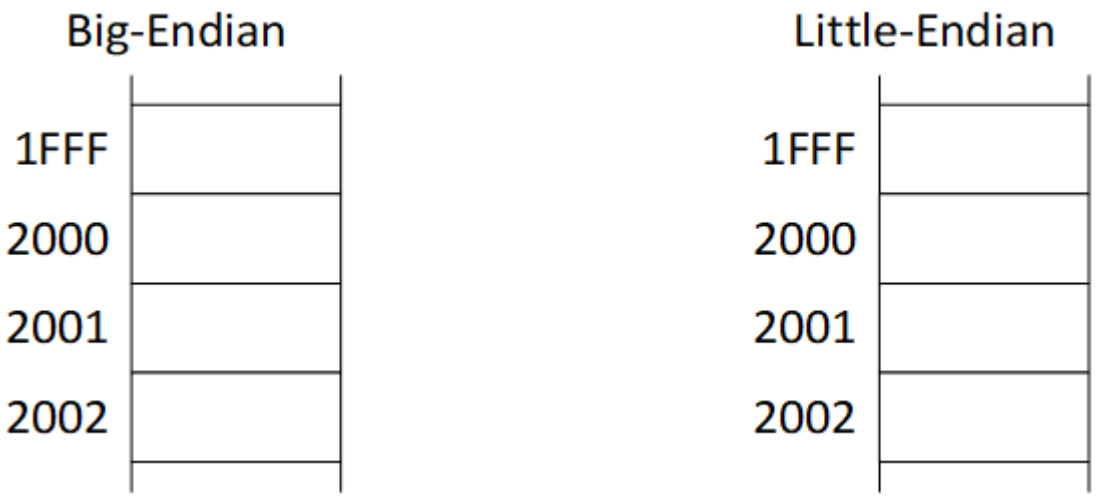
  - **Little**-endian (Intel, TI MSP430): put **the little number portion** of it **first** into the memory (the Least Significant Byte (LSB) occupies the lowest address space).



Endianness

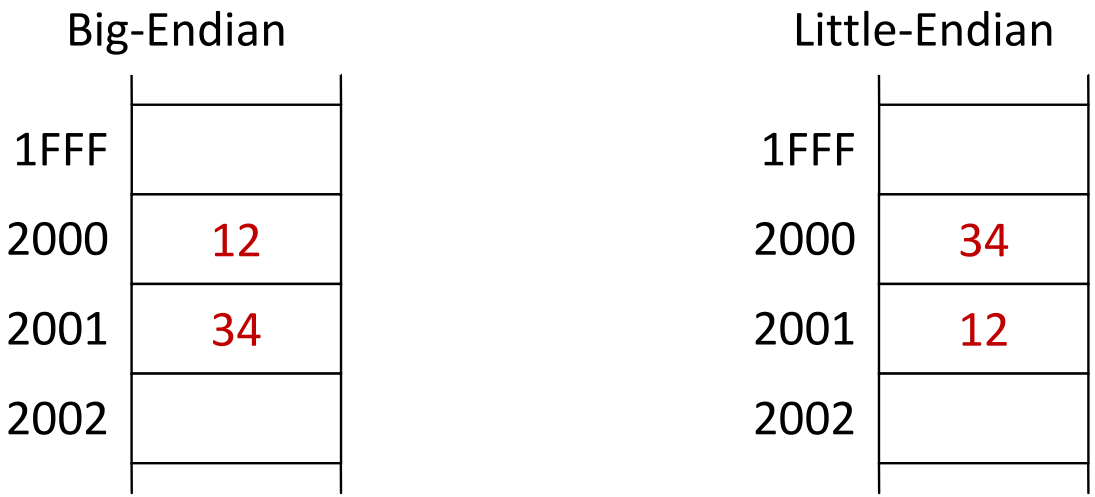
Example

- How to store the number 1234h at address 2000h?



Endianness
Example

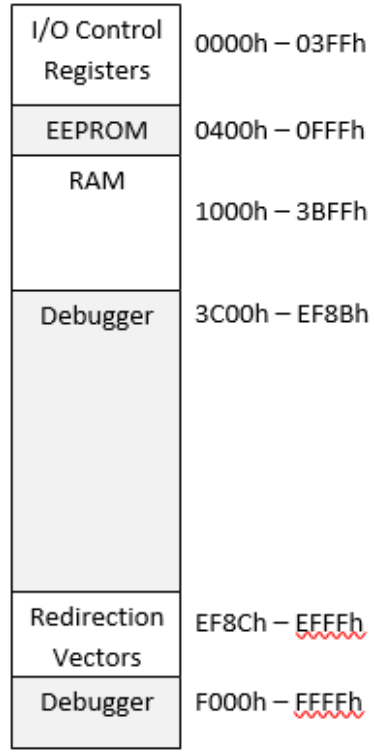
- The number 1234h stored at address 2000h



=> The memory map for the S12 which has **16-bit addresses** and **8-bit locations**.

Type of Memory

- The memory map for the S12 which has **16-bit addresses** and **8-bit locations**.



- Different ranges of addresses are mapped to different types of storage.

Instruction Set

- A list of all the operations that a processor can perform.
- A small section of the HCS12 instruction set.

Source Form	Operation	Addr. Mode	Machine Coding	Access Detail	S X H I	N Z V C
LDAA #opr8i LDAA opr8a	(M) ⇒ A Load Acc. A	IMM DIR	86 ii 96 dd	P rPf	- - - -	Δ Δ 0 -
LDAB #opr8i LDAB opr8a	(M) ⇒ B Load Acc. B	IMM DIR	C6 ii D6 dd	P rPf	- - - -	Δ Δ 0 -

- Source Form:
  - Assembly code for the instruction
- Operation
  - A brief description that explains what the instruction does.
- Addressing mode
  - It tells how the instruction uses the operand(s), if any.

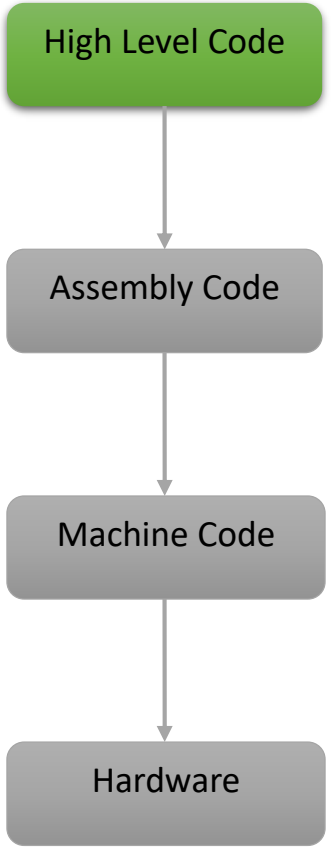
Instruction Set

Source Form	Operation	Addr. Mode	Machine Coding	Access Detail	S X H I	N Z V C
LDAA #opr8i LDAA opr8a	(M) ⇒ A Load Acc. A	IMM DIR	86 ii 96 dd	P rPf	- - - -	Δ Δ 0 -
LDAB #opr8i LDAB opr8a	(M) ⇒ B Load Acc. B	IMM DIR	C6 ii D6 dd	P rPf	- - - -	Δ Δ 0 -

- Machine Coding
  - The hexadecimal value that represents the instruction in memory.
  - Also called **the instruction format** since it shows how to convey the operation and its operands to the processor.
- Access Detail
  - Each letter stands for the internal operation performed during each clock cycle required by the operation.
  - The number of letters = the number of clock cycles taken.
- SXHINZVC: Condition Code Register
  - Δ: affected by operation, 1: set 1, and 0: set 0 after the instruction.

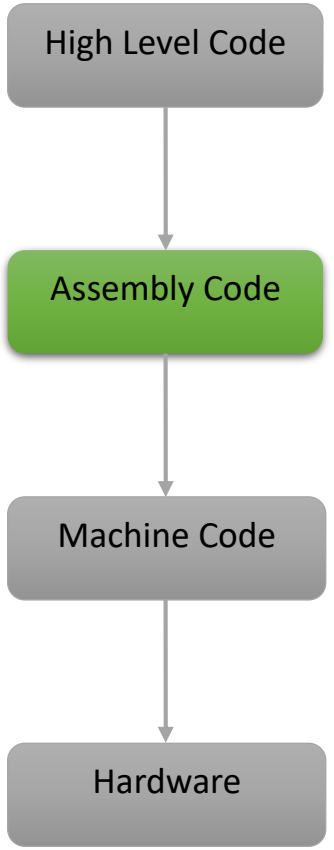
Programming Flow
High Level Code

- C, C++, Basic, Pascal, Fortran, and others
- Usually exist as a text file.
- A portion of high level may be written without regard to the specific processor that will eventually run the program
- A **compiler** converts high level code to assembly code that runs on the same processor as the compiler runs
- A **cross-compiler** runs on one type of processor and converts high level code to assembly for a different type of processor.
- High level languages do not have instructions that can access all of a microcomputer's instructions. Many programs written mainly in a high level language have sections of assembly code.
- One line in a high level language may compile into several, possibly hundreds, of lines of assembly.



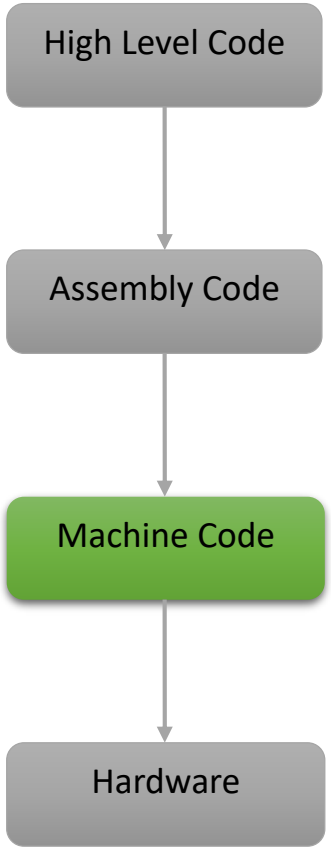
Programming Flow
Assembly Code

- A somewhat “human readable” form of the exact code that will be executed on the processor
- Usually exists as a text file
- An assembler converts assembly code to machine code that runs on the same processor as the assembler runs
- A **cross-assembler** runs on one type of processor and converts assembly code to machine code for a different type of processor
- Assembly code itself is not executed
- Assembly code is **specific** to a given type, or family, of processors
- Each line of assembly code uniquely corresponds to one instruction in machine code



Programming Flow
Machine Code

- The string of 1’s and 0’s representing the operations.
- The exact values that are loaded by the microprocessor from memory to execute the program.
- On PCs, these are executable (often .EXE) files.
- May not be executed on other types of microprocessors



Questions?

Wrap-up

What we've learned

- HSC12 Architecture Details
- Registers – Programming Model
- Memory Model – Endianness
- Programming flow

What to Come

- Addressing Mode!