EEE 204 Microcomputer Organization

Asst. Prof. Dr. Mahmut AYKAÇ

CHAPTER 3

3.1 Base Microcomputer Structure

The minimal hardware configuration of a microcomputer system is composed of three fundamental components: a Central Processing Unit (CPU), the system memory, and some form of Input/Output (I/O) Interface.

- **Central Processing Unit (CPU):** The CPU forms the heart of the microcontroller system. It retrieves instructions from program memory, decodes them, and accordingly operates on data and/or on peripherals devices in the Input-Output subsystem to give functionality to the system. (Ex: Intel Core I7, AMD Ryzen)
- **System Memory:** The place where programs and data are stored to be accessed by the CPU is the system memory. (Program Memory and Data Memory)
- **Input/Output Subsystem:** The I/O subsystem, also called *Peripheral Subsystem* includes all the components or peripherals that allow the CPU to exchange information with other devices, systems, or the external world.
- **System Buses:** The set of lines interconnecting CPU, Memory, and I/O Subsystem are denominated the system buses.

3.1 Base Microcomputer Structure

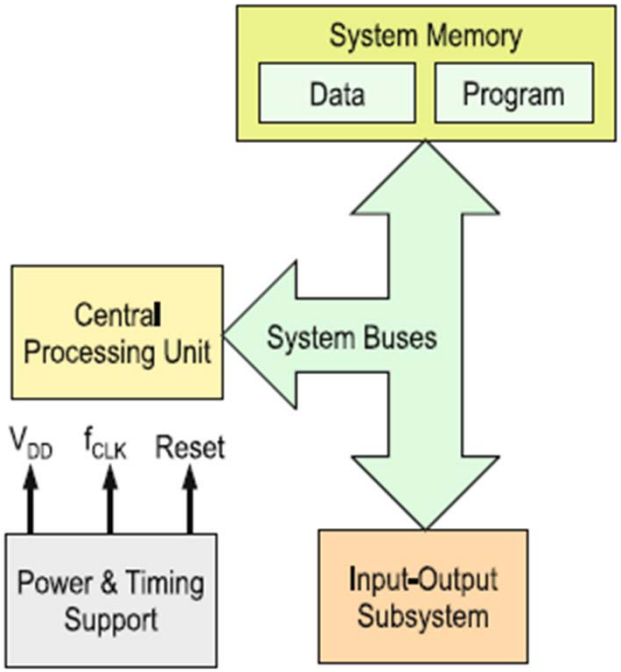


Figure. General architecture of a microcomputer system

3.2 Microcontrollers Versus Microprocessors

- **A Microprocessor Unit,** commonly abbreviated MPU, fundamentally contains a general purpose CPU in its die. The buses, memory, and I/O interfaces, are implemented externally (Ex: Intel 8085, Arm Cortex- M Series).
- **A Microcontroller Unit,** abbreviated MCU, is developed using a microprocessor core or Central Processing Unit (CPU), usually less complex than that of an MPU. This basic CPU is then surrounded with memory of both types (program and data) and several types of peripherals, all of them embedded into a single integrated circuit, or chip. This blending of CPU, memory, and I/O within a single chip is what we call a microcontroller. (Ex: PIC16F877, Atmega 328P)

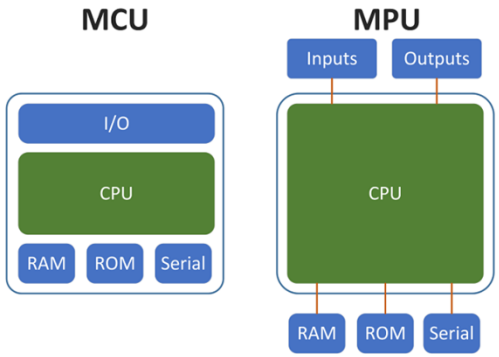


Figure. MCU vs MPU

3.2 Microcontrollers Versus Microprocessors

- **CISC (*Complex Instruction Set Computing*)** machines are characterized by variable length instruction words, i.e., with different number of bits, small code sizes, and multiple clocked-complex instructions at machine level. CISC architecture focuses in accomplishing as much as possible with each instruction, in order to generate simple programs. This focus helps the programmer's task while augmenting hardware complexity. (For ex. Intel, AMD CPUs)
- **RISC (*Reduced Instruction Set Computing*)** machines, on the other hand, are designed with focus on simple instructions, even if that results in longer programs. This orientation simplifies the hardware structure. The design expects that any single instruction execution is reduced at most a single data memory cycle when compared to the "complex instructions" of a CISC system. (For ex: Microcontroller CPUs)

Embedded systems programmers need to consider both the hardware and software issues. Hence, they need to look at the system both from a hardware point of view, the *hardware model*, as well as a software point of view, the *programmer's model*.

3.3 Central Processing Unit

The Central Processing Unit (CPU) in a microcomputer system is typically microprocessor unit (MPU) or core. The minimal list of components that define the architecture of a CPU include the following:

- **Hardware Components:**
  - An Arithmetic Logic Unit (ALU)
  - A Control Unit (CU)
  - A Set of Registers
  - Bus Interface Logic (BIL)
- **Software Components:**
  - Instruction Set
  - Addressing Modes

3.3 Central Processing Unit

- **The Arithmetic Logic Unit (ALU)** is the CPU component where all logic and arithmetic operations supported by the system are performed. Basic arithmetic operations such as addition, subtraction, and complement, are supported by most ALUs.
- Logic operations performed in the ALU may include bitwise logic operations AND, OR, NOT, and XOR, as well as register operations like SHIFT and ROTATE.
- The Control Unit (CU) governs the CPU operation working like a finite state machine that cycles forever through three states: fetch, decode, and execute, as illustrated in Figure.

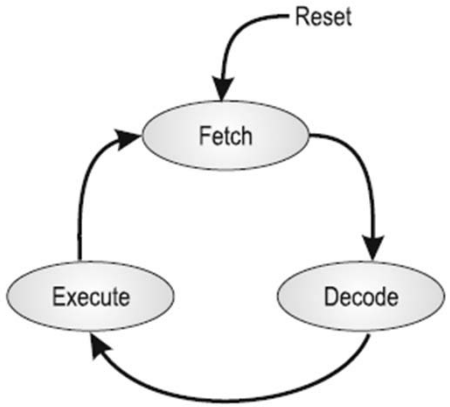
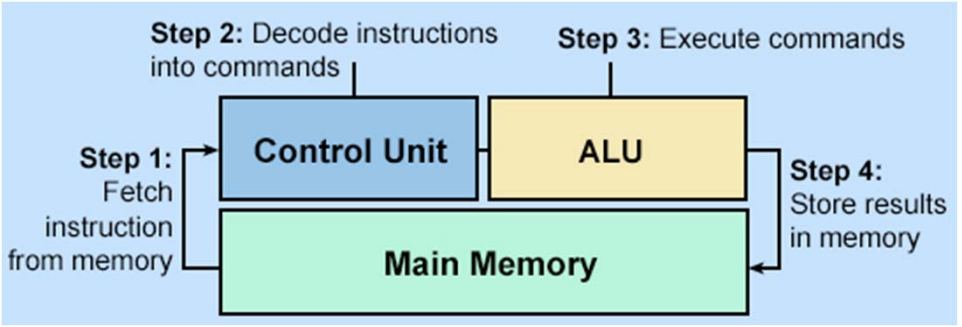


Figure. Fetch, Decode and Execute Cycle

3.3 Central Processing Unit

- **Fetch State**: During the fetch state a new instruction is brought from memory into the CPU through the bus interface logic (BIL). The *program counter* (PC) provides the address of the instruction to be fetched from memory. The newly fetched instruction is read along the data bus and then stored in the *instruction register* (IR).
- **Decoding State**: After fetching the instruction, the CU goes into a decoding state, where the instruction meaning is deciphered. The decoded information is used to send signals to the appropriate CPU components to execute the actions specified by the instruction.
- **Execution State**: In the execution state, the CU commands the corresponding CPU functional units to perform the actions specified by the instruction. At the end of the execution phase, the PC has been incremented to point to the address of the next instruction in memory.



Fetch, Decode and Execute Cycle starts with fetching an instruction from memory and ends with storing the results in the memory

3.3 Central Processing Unit

- **The Bus Interface Logic (BIL)** refers to the CPU structures that coordinate the interaction between the internal buses and the system buses. The BIL defines how the external address, data, and control buses operate.
- **CPU Registers** provide temporary storage for data, memory addresses, and control information in a way that can be quickly accessed. They are the fastest form of information storage in a computer system, while at the same time they are the smallest in capacity. Register contents is *volatile*, meaning that it is lost when the CPU is de-energized.
- **General Purpose Registers (GPR)** are those not tied to specific processor functions and may be used to hold data, variables, or address pointers as needed.

3.3 Central Processing Unit

Special Purpose (function) Registers perform specific functions that give functionality to the CPU. The most basic CPU structure includes the following four specialized registers:

- **Instruction Register (IR)** register holds the instruction that is being currently decoded and executed in the CPU.
- **Program Counter (PC)**, also **called Instruction Pointer (IP)** register holds the address of the instruction to be fetched from memory by the CPU.
- **Stack Pointer (SP)**, The *stack* is a specialized memory segment used for temporarily storing data items in a particular sequence. The operations of storing and retrieving the items according to this sequence is managed by the CPU with the stack pointer register (SP).
- **Status Register (SR)**, also called the **Processor Status Word (PSW),** or **Flag Register** contains a set of indicator bits called *flags*, as well as other bits pertaining to or controlling the CPU status.

3.3 Central Processing Unit

- **Zero Flag (ZF or Z):** Also called the zero bit. It is set when the result of an ALU operation is zero, and cleared otherwise.
- **Carry Flag (CF or C):** This flag is set when an arithmetic ALU operation produces a carry.
- **Negative or Sign flag (NF or N):** This flag is set if the result of an ALU operation is negative and cleared otherwise.
- **Overflow Flag (VF or V):** This flag signals overflow in addition or subtraction with signed numbers. A negative sum of positive operands (or vice versa) is an overflow.
- **Interrupt Flag (IF):** This flag, also called **General Interrupt Enable (GIE)**, is not associated to the ALU. It indicates whether a program can be interrupted by an external event (interrupt) or not.

3.3 Central Processing Unit

Example 3.1 *The following operations are additions performed by the ALU using 8-bit data. For each one, determine the Carry, Zero, Negative, and Overflow flags.*

01001010 +	10110100 +	10011010 +	11001010 +
01111001 =	01001100 =	10111001 =	00011011 =
0 11000011	1 00000000	1 01010011	0 11100101
↑ ↑	↑ ↑	↑ ↑	↑ ↑
C N	C N	C N	C N

Solution: *The operands have eight bits, so this length is our reference for the flags when we look at the result. The most significant bit in this group is flag* N. *The bit to the left is* C. *In hex form, these additions are, respectively,* 4Ah + 79h = C3h*;* B4h + 4Ch = 100h*;* 9Ah + B9h = 153h*; and* CAh + 1Bh = E5h. *The zero flag is set if the result is 0, discarding the carry, and the overflow flag is set if the addition of numbers of the same sign (that is, with equal most significant bit) yield a result of different sign (signaled by N). With this information we have then:*

Operation 4Ah + 79h = C3h : C = 0, N = 1, Z = 0 *and* V = 1.
Operation B4h + 4Ch = 100h : C = 1, N = 0, Z = 1 *and* V = 0.
Operation 9Ah + B9h = 153h : C = 1, N = 0, Z = 0 *and* V = 1.
Operation CAh + 1Bh = E5h : C = 0, N = 1, Z = 0 *and* V = 0.

3.3.5 MSP430 CPU Basic Hardware Structure

The MSP430 family is based on a 16-bit CPU which was introduced in the early models of the series 3xx.

Registers: They are high-speed memory units, which are in CPU, that can also be accessed by the CPU for fast operations.

MSP430 Registers: There are sixteen 16-bit registers in the MSP430 CPU named R0, R1…, R15. Registers R4 to R15 are of the general purpose type. The specialized purpose registers are:

- Program Counter register, named **R0** or PC.
- Stack Pointer Register, named **R1** or SP.
- Status Register, with a dual function also as Constant Generator. It is named **R2**, SR or CG1.
- Constant Generator, named CG2 or **R3**.

3.3.5 MSP430 CPU Basic Hardware Structure

- **Status Register** The SR register has the common flags of Carry (C), Zero (Z), Sign or Negative (N), overflow (V) and general interrupt enable (GIE). It contains in addition a set of bits, CPUOFF, OSCOFF, SCG1 and SCG0, used to configure the CPU, oscillator and power mode operations.
- **Arithmetic-Logic Unit:** TheMSP430 CPU ALU has a 16-bit operand capacity; the CPUX has 16- or 20-bit operand capacity. It handles the arithmetic operations of addition with and without carry, decimal addition with carry.

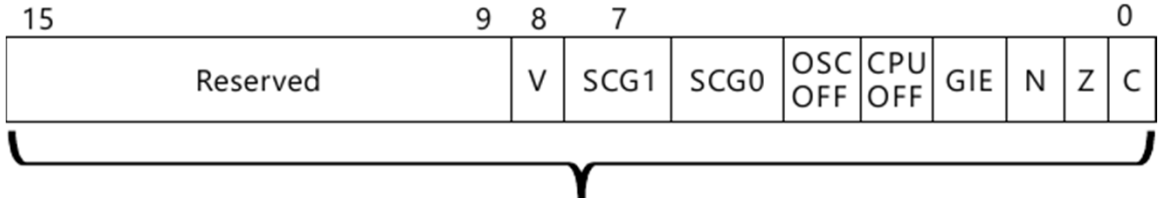


Figure. Status Register (SR) of MSP430

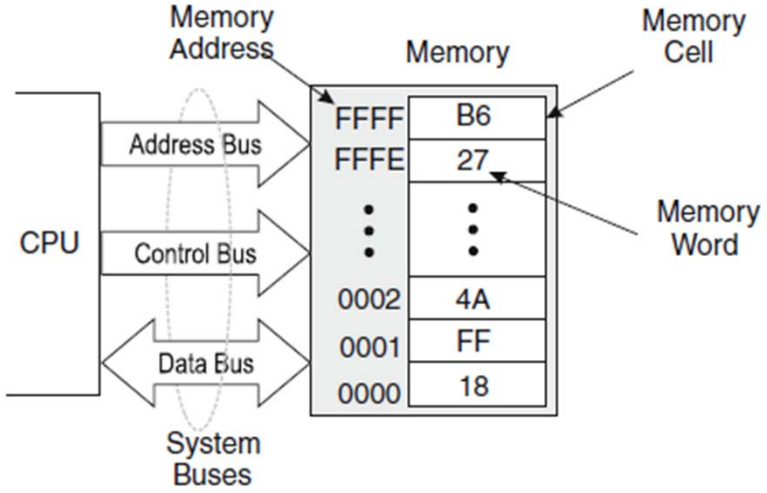
3.4 System Buses

Memory and I/O devices are accessed by the CPU through the system buses. A bus is simply a group of lines that perform a similar function. Each line carries a bit of information and the group of bits may be interpreted as a whole. The system buses are grouped in three classes: **address bus, data bus, and control bus.**

- The set of lines carrying data and instructions to or from the CPU is called the **data bus.**
- The CPU interacts with only one memory register or peripheral device at a time. Each register, either in memory or a peripheral device, is uniquely identified with an identifier called **address**. The set of lines transporting this address information form the **address bus. The width of the address bus determines the size of the largest memory space that the CPU can address.** (Ex: 64-bit Intel or AMD CPUs)
- The **control bus** groups all the lines carrying the signals that regulate the system activity. Unlike the address and data buses lines which are usually interpreted as a group (address or data), the control bus signals usually work and are interpreted separately.

3.5 Memory Organization

The memory subsystem stores **instructions** and **data**. Memory consists of a large number of hardware components which can store one bit each. These bits are organized in *n*-bit words, working like a register, usually referred to as cell or location.



There is one byte in one address (byte addressing)
Address: Data
0000: 18
0001: FF
0002: 4A
……..: ….
FFFE: 27
FFFF: B6

Figure. Memory structure of 64kB memory

3.5 Memory Organization

- Hardware memory is classified according to two main criteria: **storage permanence** and **write ability**.
- From the storage permanence point of view, the two basic subcategories are the **nonvolatile** and **volatile** groups.

Storage	Memory	In-system Writable	Comments
Nonvolatile	Masked ROM	No	Non programmable
	OTPROM	No	One time programmable with programming device
	EPROM	No	Erasable and programmable with external device
	EEPROM	Yes	Slow to erase/write. Not advisable to write during program execution. Requires higher voltage.
	Flash	Yes	Similar to EEPROM
	FRAM	Yes	Fast to write at low voltage
Volatile	Static RAM	Yes	Fastest to write/read
	DRAM	Yes	Fast to write/read

Figure. Memory types

3.5 Memory Organization

A microcomputer includes two differentiable types of memory depending on the kind of information they store: **Program Memory** and **Data Memory.**

- **Program Memory (Flash)**, as inferred by its name, refers to the portion of memory that stores the system programs in a form directly accessible by the CPU. A *program* is a logical sequence of instructions that describe the functionality of a computer system.
- **Data Memory (RAM)** is used for storing variables and data expected to change during program execution. Therefore, this type of memory should allow for easily modifying its contents. Most embedded systems implement data memory in RAM (Random Access Memory).

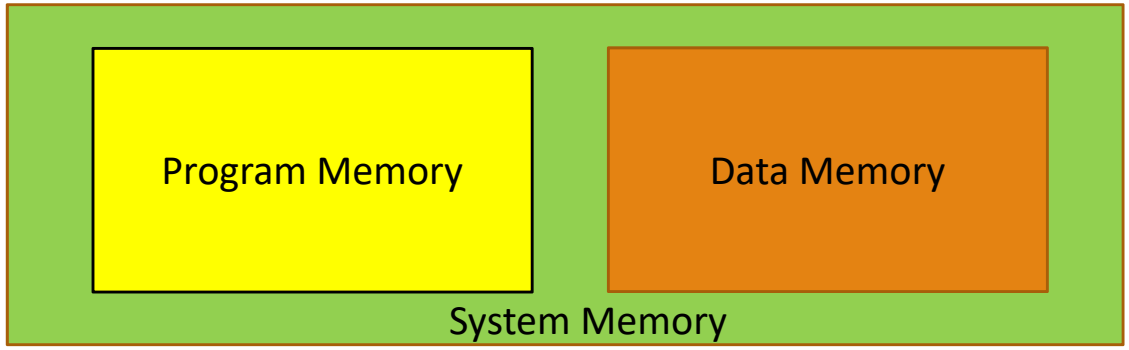


Figure. System Memory (Program Memory and Data Memory)

3.5.4 Von Neumann and Harvard Architectures

- Systems with a single set of buses for accessing both programs and data are said to have a **Von Neumann** architecture or **Princeton** architecture.
- An alternate organization is offered by the **Harvard Architecture**. This topology has physically separate address spaces for programs and data, and therefore uses separate buses for accessing each. Data and address buses may be of different width for both subsystems.

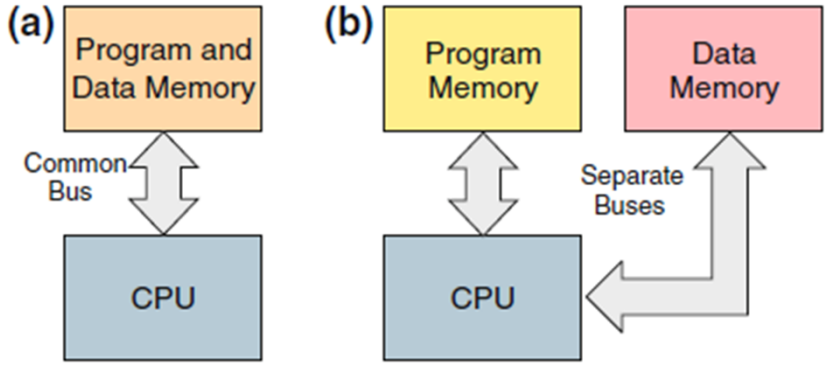


Figure. (a) Von Neumann arch., (b) Harvard arch.

Texas Instruments MSP430 series MCUs employ Von Neumann architecture!!

3.5.6 Memory Map

A **memory map** is a model representation of the usage given to the addressable space of a microprocessor based system. It is an important tool for program planning and for selecting the convenient microcontroller for our application. As implied by its name, the memory map of a microcomputer provides the location in memory of important system addresses.

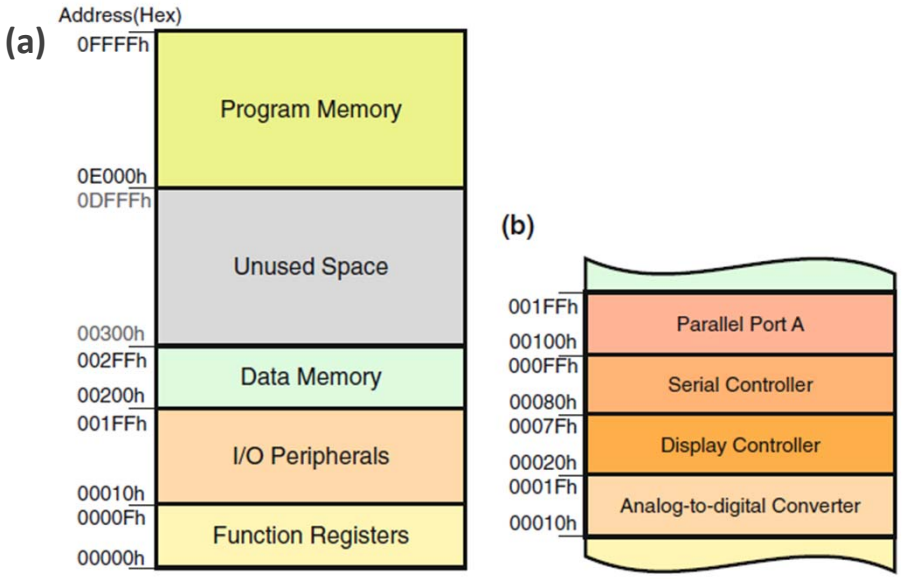


Figure. Example memory map for a microcomputer with a 16-bit address bus. **(a)** Global memory gap, **(b)** Partial memory map for I/O peripherals

3.5.7 MSP430 Memory Organization

- The address bus width depends on the microcontroller model. All models are based on the original 16-bit address bus with an address space of 64KBytes, called simply the MSP430 architecture. The extended MSP430X architecture has a 20-bit address bus with an address space of 1Mbyte.
- The amount of RAM and Flash or ROM (Flash memory is also a category of ROM memory) depends on the model. RAM memory, which may start with only 128 bytes of capacity, usually starts at address 0200h, and ends depending on the model. Similarly, in the 16-bit model, the Flash/ROM memory ends at address 0FFFFh but the start depends on the capacity.

3.6 I/O Subsystem Organization

- The I/O subsystem is composed by all the devices (peripherals) connected to the system buses, other than memory and CPU. The I/O designation is collectively given to devices that serve as either input, output, or both in a microprocessor-based system, and also includes special registers or devices used to manage the system operation without external signals.
- Examples of input devices include switches and keyboards, barcode readers, position encoders, and analog-to-digital converters (ADC).
- Output devices include LEDs, displays, buzzers, motor interfaces, and digital-to-analog converters (DAC).

3.6.2 Parallel Versus Serial I/O Interfaces

- In microcomputer-on-a-chip systems, most peripherals are connected to the data bus via a parallel interface, i.e., all bits composing a single word are communicated simultaneously, requiring one wire per bit. But I/O ports interacting with devices external to the system, may connect via parallel or serial interfaces. The ports are then referred to as **parallel I/O ports** and **serial I/O ports**. Serial interfaces require only one wire to transfer the information, sending one bit at a time.

Figure 1. Parallel Communication

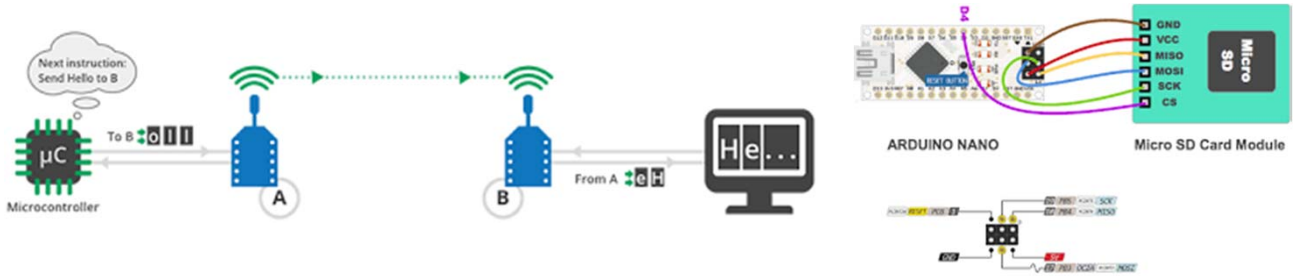


Figure 2. Serial Communication examples, UART and SPI

3.6.5 MSP430 I/O Subsystem and Peripherals

MSP430 microcontroller models, as most microcontrollers, usually have more peripherals than necessary for a given application. Therefore, to minimize resources, several modules may share pins.

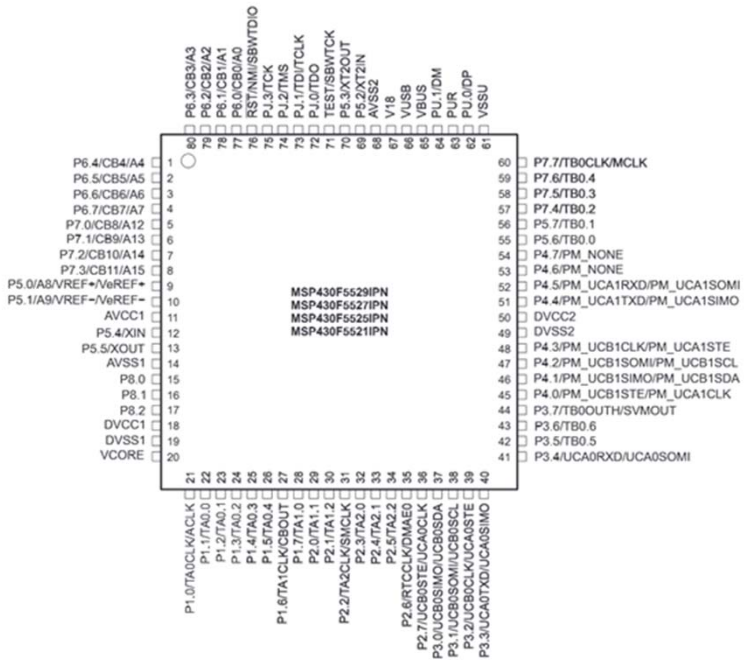


Figure 1. MSP430F5529 Pinout

Most of the pins are shared as seen in the figure. For ex: Pins 37, 38, 39 and 40 are shared to save space for the chip on the target board. Otherwise, chip would cover more space.

3.7.2 Machine Language and Assembly Instructions

To avoid using binary or hexadecimal representation for instruction words, scientists and engineers devised *high level* and *assembly* languages. Java, C, and Basic belong to the first group. Instructions in assembly language, the first to appear, on the other side, are the same as machine language. That is, each assembly instruction is associated with one, and only one, machine language instruction.

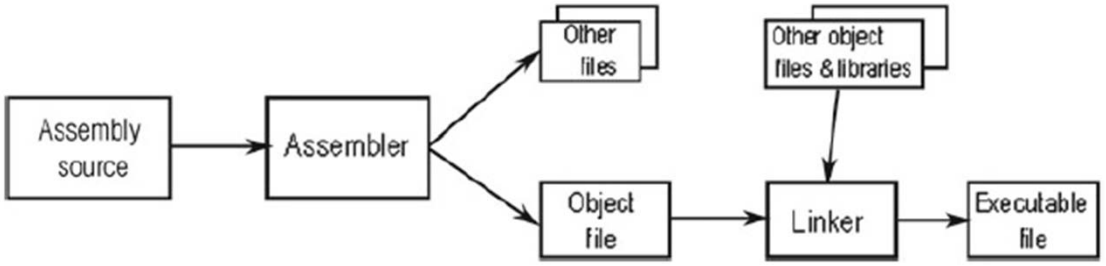


Figure. Basic Assembly process

Machine language	Assembly language
480D	`mov R8,R13`
5079 006B	`add.b #0x6B,R9`
23F1	`jnz 0x3E2`
1300	`reti`

Machine language	Assembly language
C8 34	`EORB #$34`
C0 07	`LDD #7`
7E 10 00	`JMP $1000`
5C	`INCB`

3.7.4 The Stack and the Stack Pointer

The **stack**, a specialized volatile memory segment used for temporarily storing data, is managed with the *Stack Pointer* (SP), both by programming or by the CPU itself. To store data in this segment using a stack transaction is to **push**. Retrieving a datum from the stack is to **pop**, or to **pull**.

3.7.5 Addressing Modes

Addressing modes can be defined as the way in which an operand is specified within an instruction so as to indicate where to find the data with which the operation is executed. The addressing mode is denoted using a specific syntax format, proper of the microcontroller family. Instructions with implicit operands are said to use **Implicit Addressing Modes**.

In general, the data to be used or stored in a transfer or in an arithmetic or logic instruction can be located in only one of the following possible places:

1. It may be explicitly given

2. It may be stored in a CPU register

3. It may be stored at a memory location, or

4. It may be stored in an I/O port or peripheral register

3.7.5 Addressing Modes

- Addressing modes in computer architecture refer to the techniques and rules used by processors to calculate the effective memory address or operand location for data operations.
- Simply, addressing modes tell us how to use the instructions and operands (source and destination)

Assembly	RTN (**Register Transfer Notation**)	Comment
`mov src,dest`	dest ← src	Copy or load source to destination
`add src,dest`	dest ← dest + src	Add source to destination
`sub src,dest`	dest ← dest − src	Subtract source from destination
`and src,dest`	dest ← dest .AND. src	Bitwise AND source to destination
`xor src,dest`	dest ← dest .XOR. src	Bitwise XOR source to destination
`cmp src,dest`	dest − src	Compare does not affect dest., only flags

Table. Addressing mode examples

3.9 Introduction to Interrupt and Reset

The topic of interrupts and resets involve both hardware and software subjects, but it is also closely related to how a CPU operates.

- An **Interrupt** is a response by the processor to an event that needs attention from the software. An interrupt condition alerts the processor and serves as a request for the processor to interrupt the currently executing code when permitted, so that the event can be processed in a timely manner.
- A **Reset** is an asynchronous signal that when fed to an embedded system causes the CPU and most of the sequential peripherals to start from a predefined, known state.

3.11 The TI MSP430 Microcontroller Family

All MSP430 family members are developed around a 16-bit RISC CPU with a Von-Neumann architecture. The assortment of peripherals and features in each MSP430 device varies from one series to another, and within a series from one family member to another. Common features to devices include:

- **Standard 16-bit Architecture:** All devices share the same core 16-bit architecture and instruction set. The 20-bit CPUX registers are also based on this architecture.
- **Different Ultra Low-Power Operation modes:** The devices can operate with nominal supply voltages from 1.8V to 5.5V. The nominal operating current at 1MHz ranges from 0.1 µA to 400 µA, depending on the supply voltage. The wake-up time from standby mode is 6 µS.
- **Flexible and Powerful Processing Capabilities:** The programmer's model provides seven source-address modes and four destination-address modes with 27 core instructions. Extensive interrupt capability with prioritized, nested interrupts with unlimited depth level. A large register file with RAM execution capability, table processing, and hex-to-decimal conversion modes.

3.11 The TI MSP430 Microcontroller Family

- Extensive, memory-mapped peripheral set including: a 14-bit SAR A/D converter, multiple timers and PWM capability, slope A/D conversion (all devices); integrated USART, LCD driver, Watchdog Timer, multiple I/O lines with interrupt capability, programmable oscillator, 32-kHz crystal oscillator (all devices).
- Versatile device options include:
  - Masked ROM
  - OTP and EEPROM models with wide temperature range of applications
  - Models with ferroelectric memory
  - Up to 64K addressing space for the MSP430 and 1M for the MSP430X
  - Memory mixes to support all types of applications.
- JTAG/debugger element, used for debugging embedded systems directly on the chip.

3.12.2 Programming and Debugging Tools

- Among the programming and debugging tools we have MSP430 simulators, C compilers and assemblers, linkers, and real time operating systems (RTOS).
- In the content of the lecture, we use **Code Composer Studio (CCS).**

Site	Simulator	C compiler	Assembler	Linker
CCS	X	X	X	X
IAR	X	X	X	X
mspgcc		X	X	X
naken	X		X	X
pds-430	X	X	X	
cdk4msp	X			
mcc-430		X		

Table. Programming and debugging tools for MSP430