Microprocessors for Embedded Systems

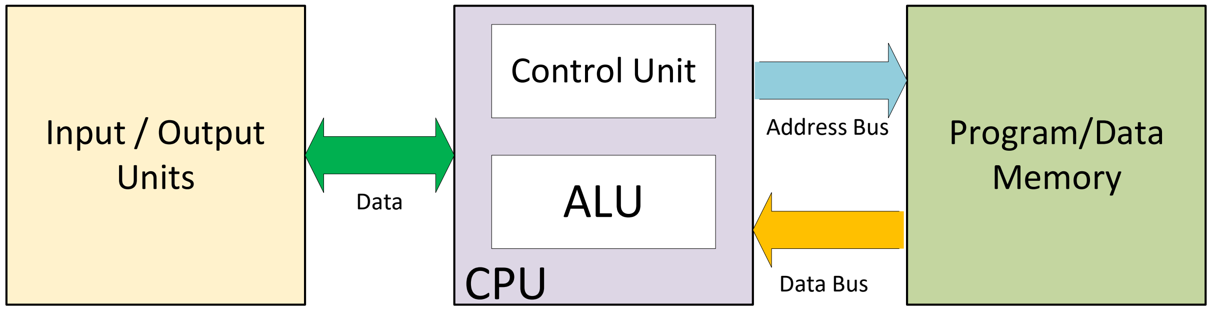
Luis F. González P.
ITESM GDA

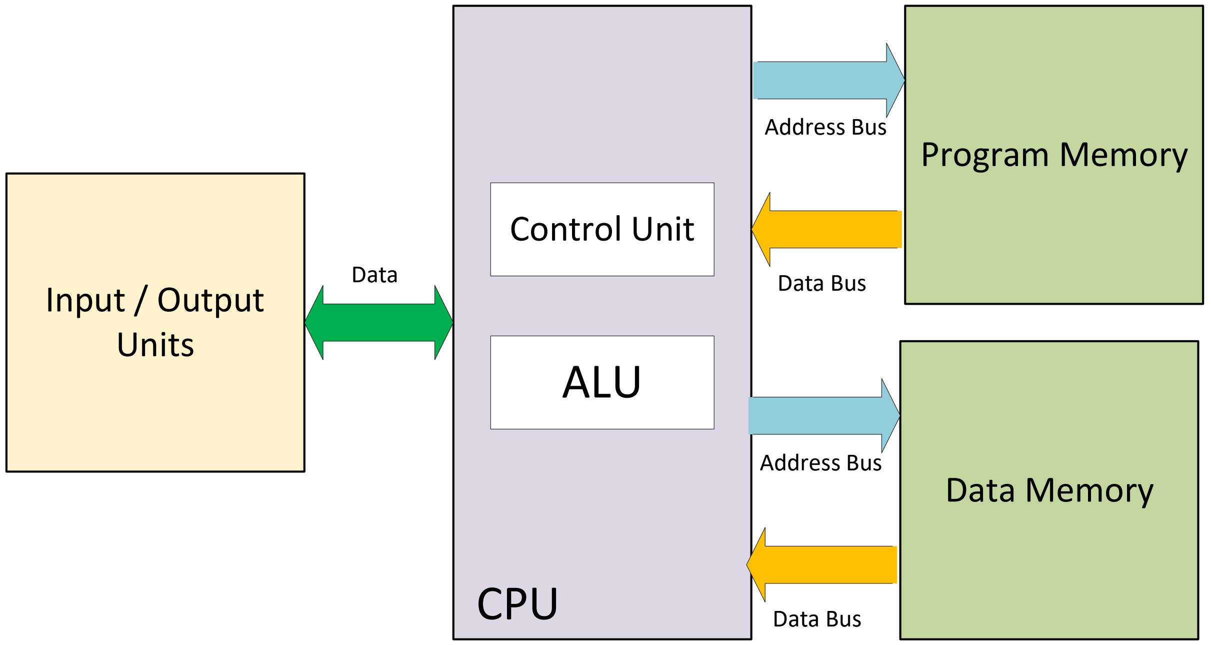
What is an embedded computer?

- An embedded computer is:
  - A digital computer that
  - Reads binary instructions from its memory,
  - Accepts binary data (through ADCs or any other interfacing)
  - Processes this data according to those instructions, and
  - Provides results as output (through DACs or any other output interfacing)
- As with any digital computer, an embedded computer contains five essential parts of units:
  - Arithmetic Logic Unit (ALU)
  - Control Unit
  - Memory Unit
  - Input Unit
  - Output Unit

In other words, an embedded computer is a combination of a microprocessor-based hardware and the suitable software to undertake a specific task

Embedded computer main components





Embedded computer main components

- ALU
  - Performs arithmetic and logical operations on data
  - The type of operations to be performed is dictated by the **Control Unit**
  - All other elements of the embedded computer are there to bring data into the ALU and to take data back out
  - Data is presented to the ALU in registers
  - Data can come from the **Memory Unit** or the **Input Unit**
  - Results of the operations performed in the ALU can be transferred either to the **Memory Unit** of the **Output Unit**

Embedded computer main components

- Control Unit
  - It is the orchestra director, responsible for keeping each member in proper synchronization
  - It contains logic and timing circuits that generate the proper signals to execute each instruction in a program
  - It **fetches** an instruction from memory by sending an address and read command
  - This instruction is **decoded** to determine which instruction/operation is being called for
  - Once decoded, the Control Unit generates the required signals to all the Units in order to **execute** the specified operation

Embedded computer main components

- Memory Unit
  - Mixture of RAM and ROM, HDD, etc.
  - It stores the instructions that the computer will perform (**program**)
  - It stores the data that are to be operated on by the program
- Input Unit
  - Devices used to take information and data that are external to the embedded computer
  - Data are put into memory or ALU
- Output Unit
  - Devices used to transfer data and information from the computer to the outside world

What is a Microprocessor?

- The microprocessor is the central processing unit (CPU) of an embedded computer
- It contains the necessary circuitry of the control and ALU
- It is the heart of every computer. It performs a number of crucial functions, such as:
  - Providing timing and control signals for all the elements of the embedded comptuter
  - Fetching instructions and data from memory
  - Decoding instructions
  - Transferring data to and from memory and I/O devices
  - Performing arithmetic and logic operations
  - Responding to I/O generated control signals sich as RESET and INTERRUPT

What is a Microprocessor?

- A microprocessor can be divided into three parts
  - ALU
  - Register Unit
  - Control Unit
- ALU: already explained
- Register Unit: A bunch of registers that are used to store data temporarily during the execution of a program
- Control Unit: Provided the necessary timing and control signals to all the operations in the embedded computer. It controls the flow of data between the microprocessor and peripherals

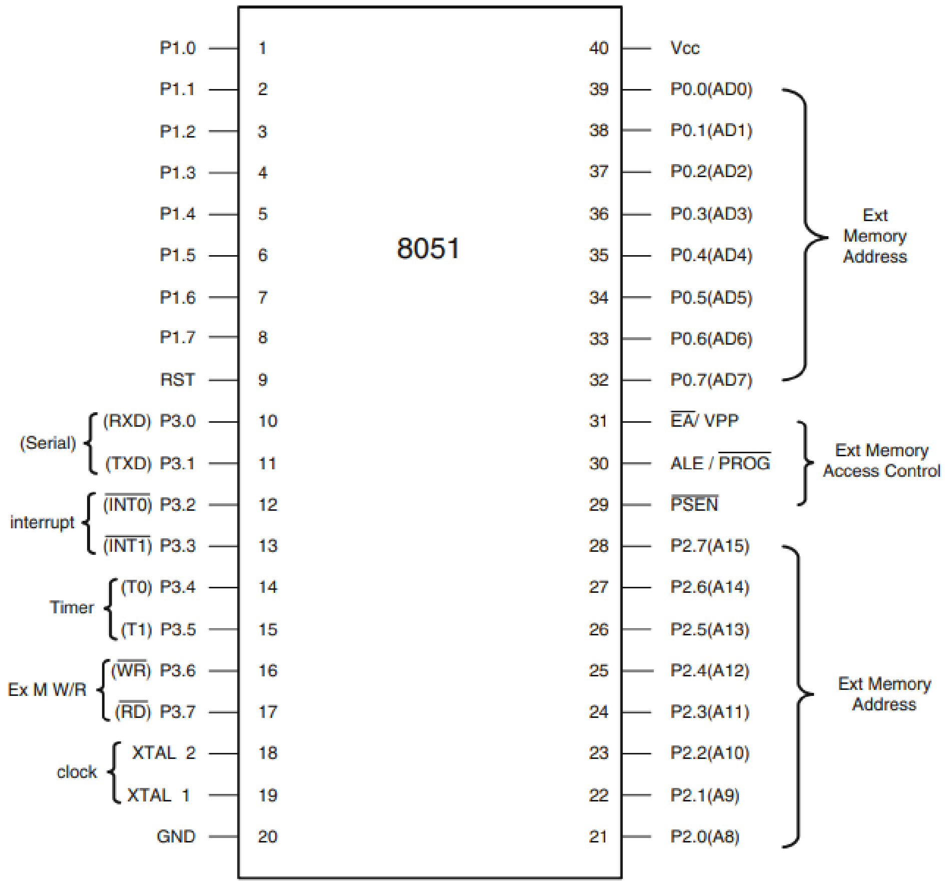
A classic example – 8051 uC

- Very popular general purpose uC widely used for small scale embedded systems
- Features
  - 8-bit data bus
  - 16-bit address bus (64K byte code memory space and a separate 64K data memory space)
  - 4K on-chip ROM memory
  - 128 bytes of internal RAM organized in Harvard Architecture
  - Two timers/counters (16-bit)
  - Serial port

A classic example – 8051 uC

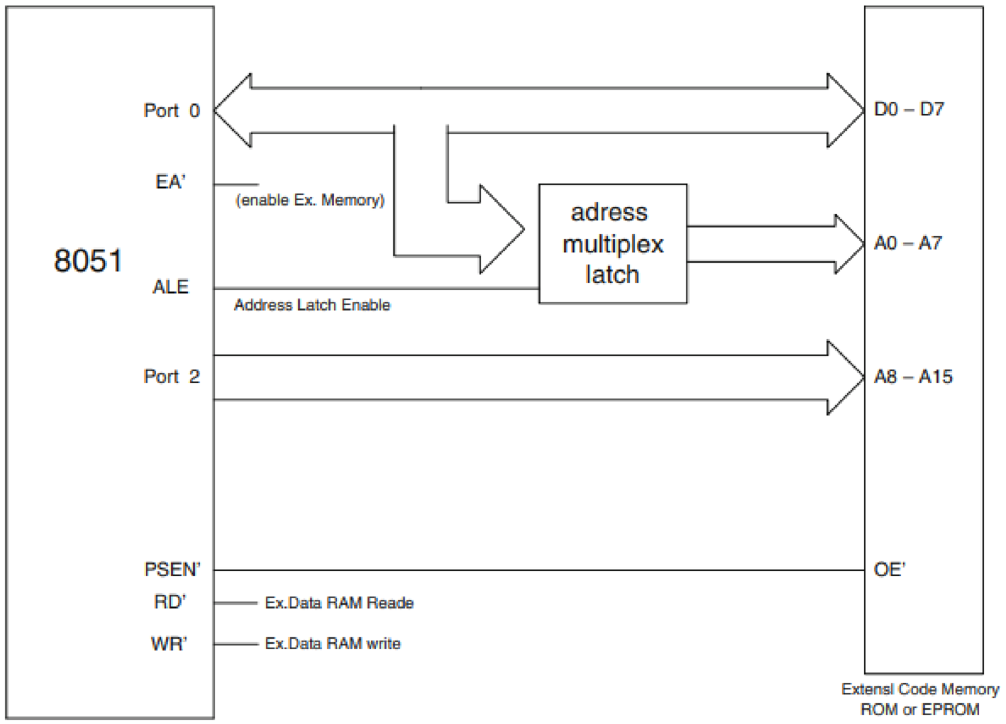
- Features
  - 4 general purpose parallel I/O ports
  - Interrupt control logic with 5 sources of interrupts
  - Special Functions Registers (SFR) such as accumulator, B register and other control registers
  - 34 8-bit general purpose registers
  - ALU performs one 8-bit operation at a time

8051 Chip Pins



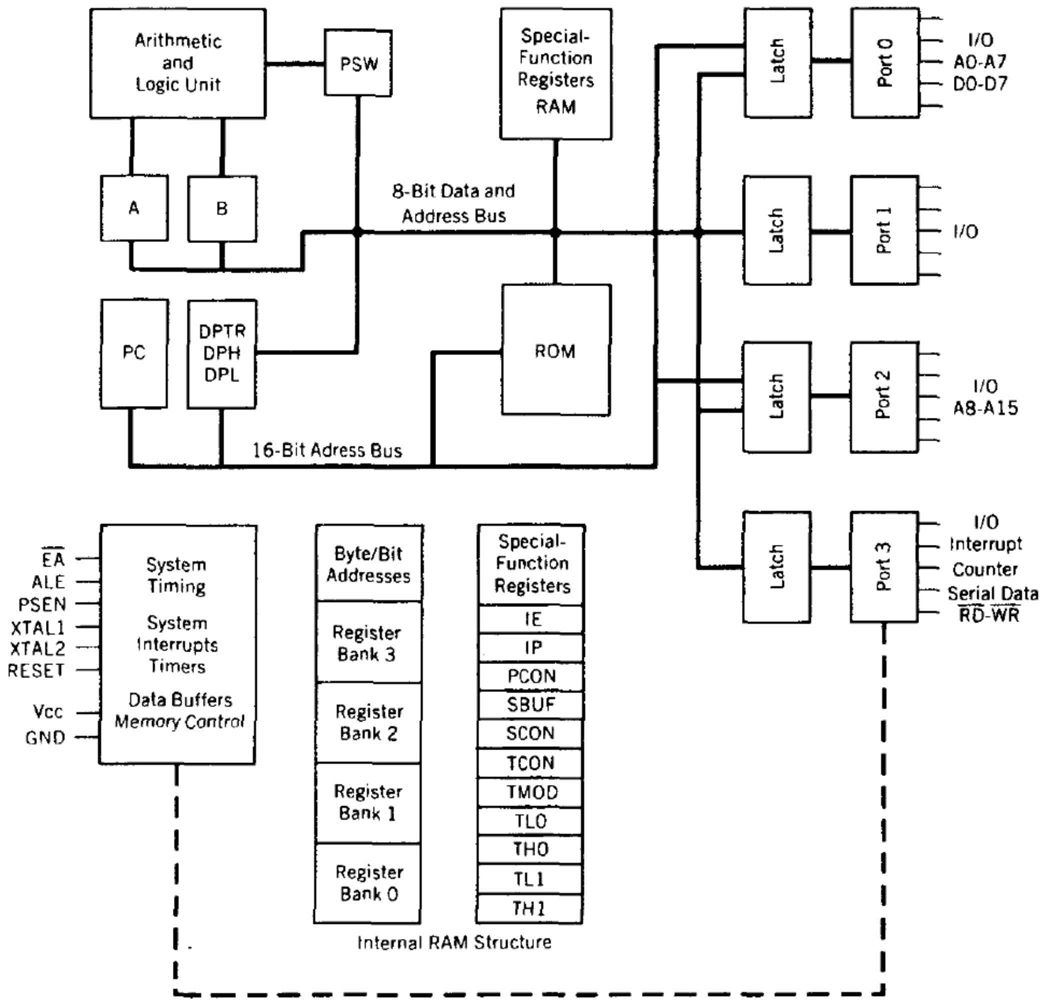
- Four I/O ports
- XTAL pins for clock
- Timer pins for timing controls
- Internal and external data and code memory access controls (EA’, ALE, PSEN, WR, RD)
- Reset pin for reboot

8051 – External Code and Data Memory Connections



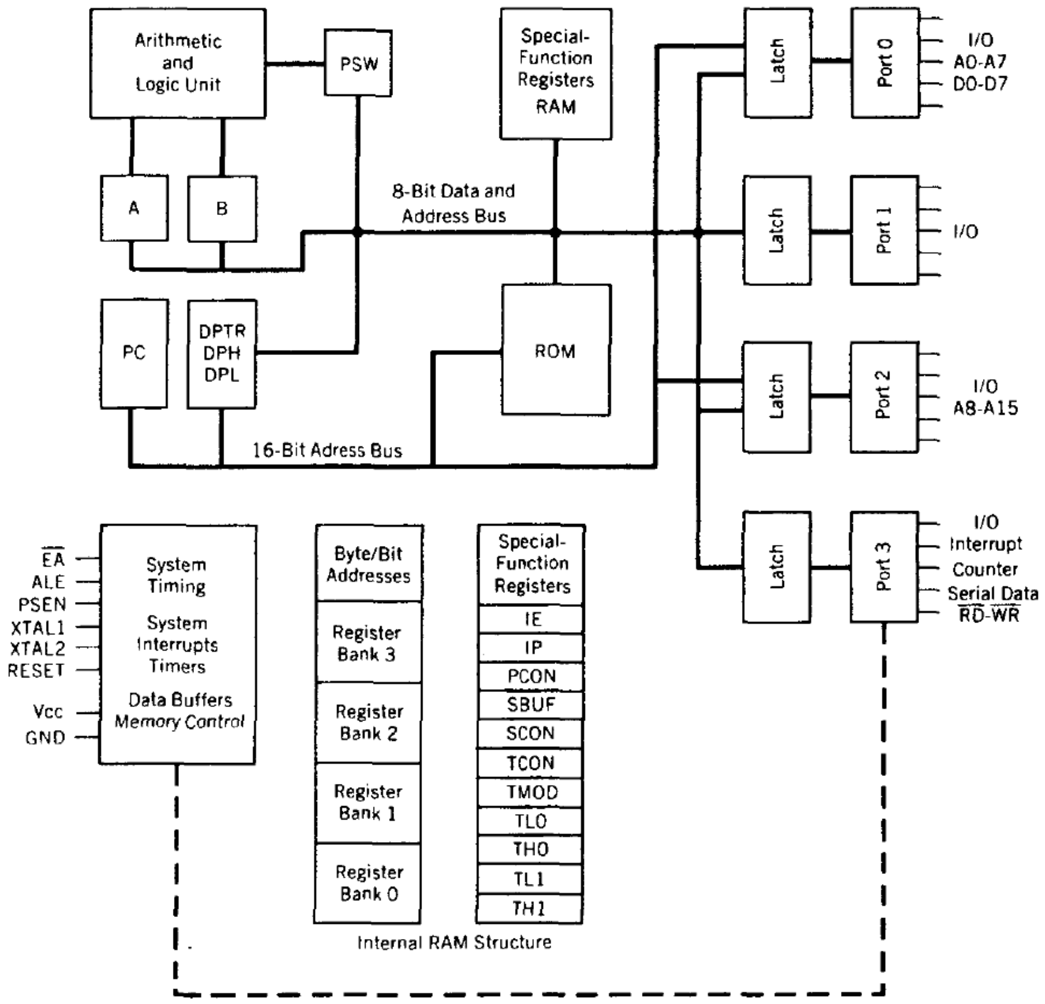
- EA' controls access to internal/external memory
  - 0 is for internal access
  - 1 is for external access
- PSEN' (Program Store Enable) is for reading external code memory when it is low (0) and EA is low (0)
- ALE (Address Latch Enable) activates port 0 joined with port 2 to provide 16-bit external address bus
- ALE multiplexes port P0
  - 1 for latching address on P0 as A0-A7 in the 16-bit address bus
  - 0 for latching P0 as data I/O

8051 Block Diagram



- Program Counter (PC) holds the code memory location of the next instruction
- CPU fetches instructions from the code memory into the Instruction Register (IR), analyzes the opcode, updates de PC to the next instruction, fetches operands from data memory if necessary and performs the operation in the ALU

8051 Block Diagram



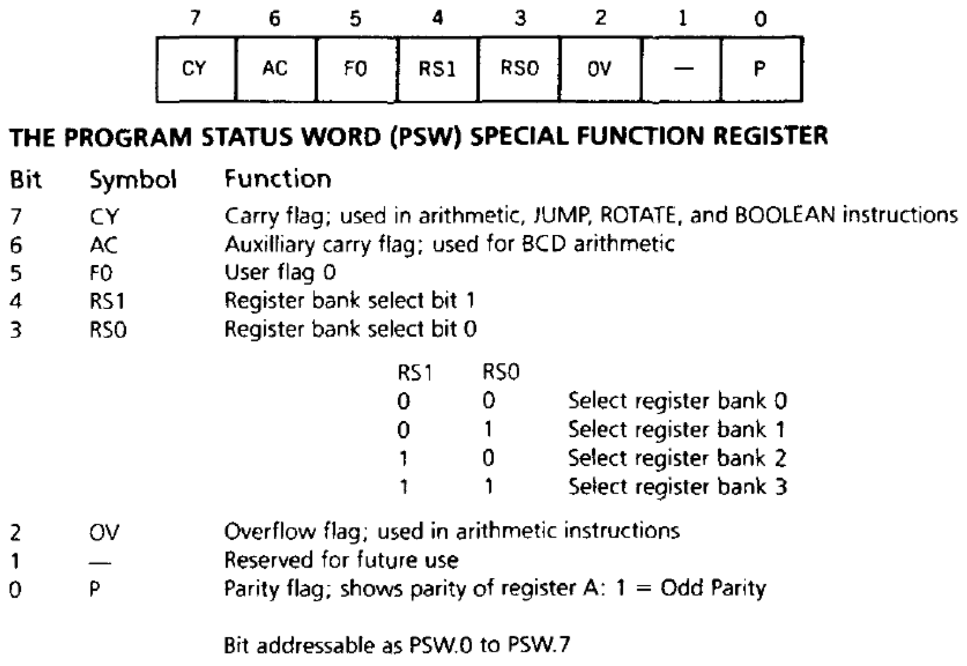
- The B register is a register for mulitiplication and division
- Immediate results are stored in A register (Acc)
- PSW (Program Status Word) is updated depending on the status of the operation result
- DPTR (Data Pointer) for indirect memory data access

8051 – Program Counter and Data Pointer

- Program instruction bytes are fetched from memory that are addressed by the PC
- Program ROM may be on-chip (addresses 0000h – 0FFFh), external to the chip for addresses that exceed 0FFFh or totally external (0000h – FFFFh)
- The PC is automatically incremented after every instruction byte is fetched and may also be altered by certain instructions
- The Data Pointer is made up of two 8-bit registers, DPH and DPL. It is used to furnish memory addresses for internal and external code access and external data access

8051 – Flags and Program Status Word

- Flags are 1-bit registers provided to store the results of certain program instructions
- Other instructions can test the condition of the flags and make decisions based upon the flag states
- For the flags to be conveniently addressed, they are grouped in the Program Status Word (PSW) and power control (PCON) registers



7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	OV	—	P

THE PROGRAM STATUS WORD (PSW) SPECIAL FUNCTION REGISTER

Bit	Symbol	Function
7	CY	Carry flag; used in arithmetic, JUMP, ROTATE, and BOOLEAN instructions
6	AC	Auxilliary carry flag; used for BCD arithmetic
5	F0	User flag 0
4	RS1	Register bank select bit 1
3	RS0	Register bank select bit 0

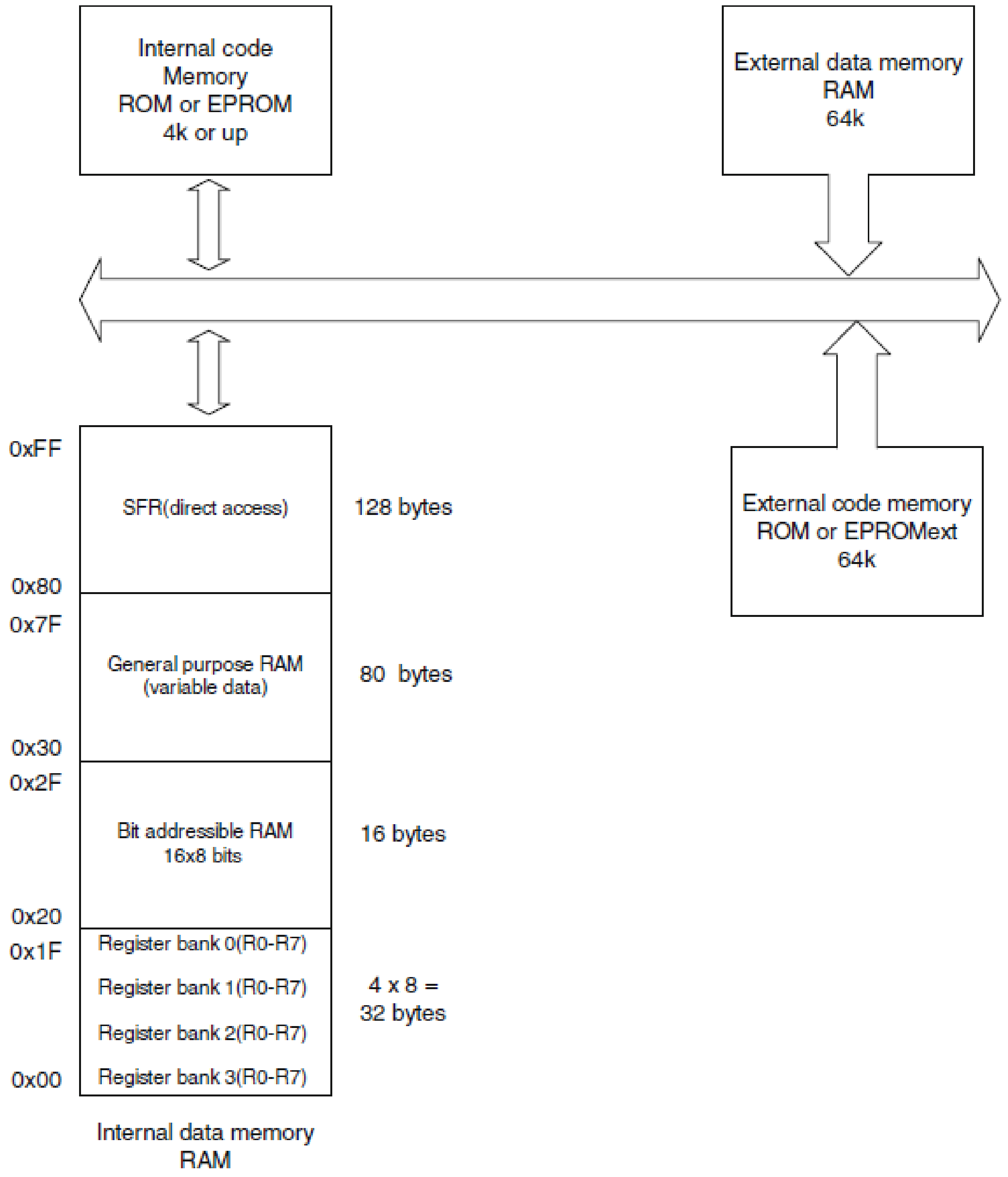
RS1	RS0	
0	0	Select register bank 0
0	1	Select register bank 1
1	0	Select register bank 2
1	1	Select register bank 3

Bit	Symbol	Function
2	OV	Overflow flag; used in arithmetic instructions
1	—	Reserved for future use
0	P	Parity flag; shows parity of register A: 1 = Odd Parity

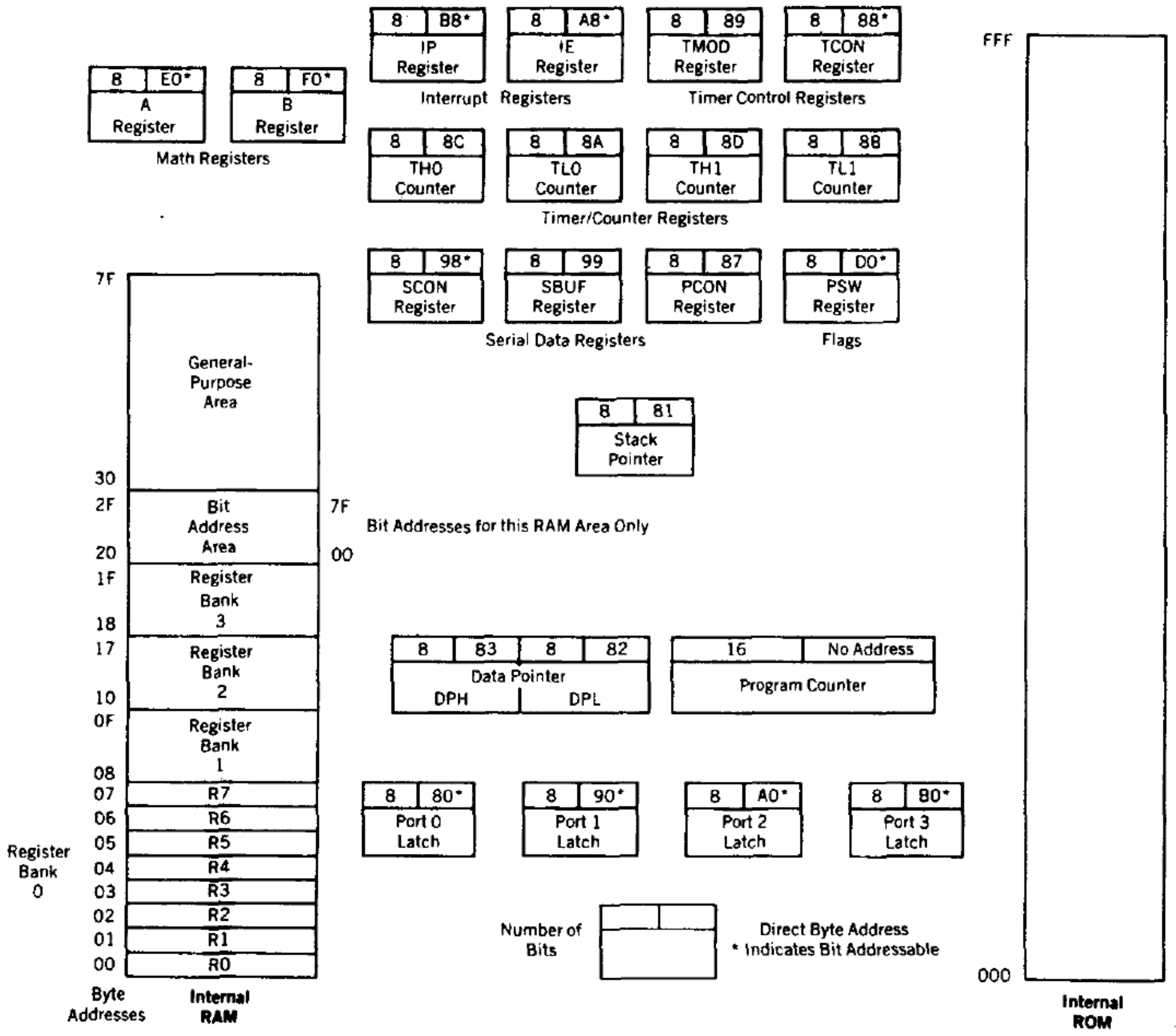
Bit addressable as PSW.0 to PSW.7

8051 – Memory

- Memory is organized in a Harvard Architecture
- 256 bytes of internal RAM
- Only 128 accessible for general use
- Second 128 bytes are used to store Special Function Registers

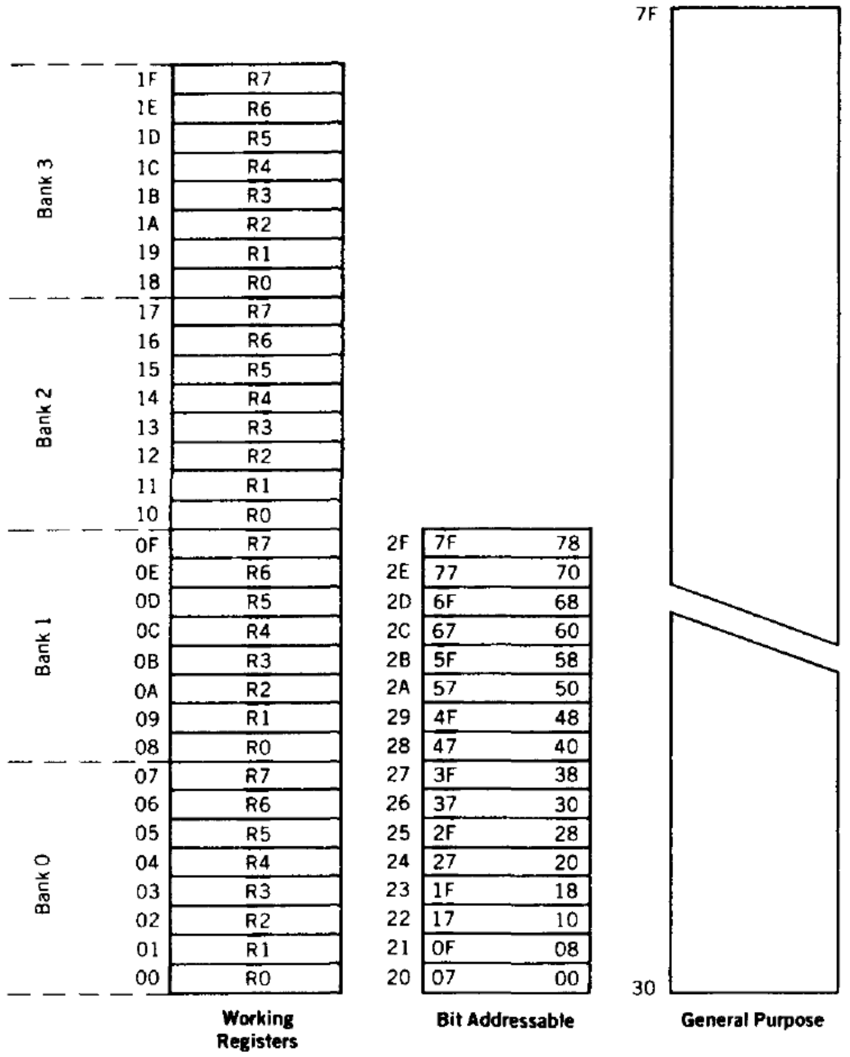


8051 – Memory Organization / Programming Model



- 256 bytes of internal RAM
- Only 128 accessible for general use
- Second 128 bytes are used to store Special Function Registers (controlling ports, timers, interrupts, serial comms, etc.)

8051 – Memory Organization / Programming Model

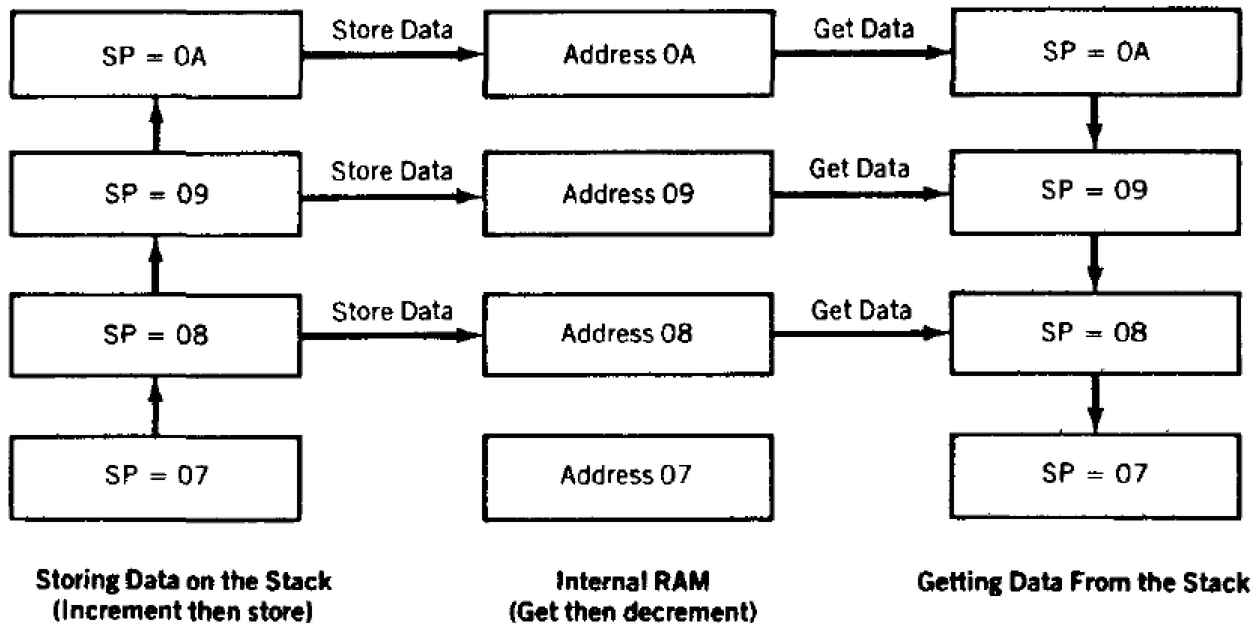


- Lowest 32 bytes are reserved for 4 general register banks, i.e., 32 8-bit registers divided into 4 banks
- Banks are numbered 0 – 3 and registers R0 – R7
- Each register can be addressed by name or by RAM address
- Bits RS0 and RS1 of PSW indicate which bank is in use
- Bank 0 is selected upon reset
- Advantages of banks:
  - Saves time on context switches for interrupted programs to store and recover their status
  - Otherwise, push and pop stack operations are needed to save the current state and to recover ir after the interrupt is over

8051 – Stack Pointer

- The stack is an area of internal RAM that is used in conjunction with certain opcodes to store and retrieve data quickly
- The SP register is used to hold an internal RAM address that is called the "top of the stack"
- The address in the SP is the location in internal RAM where the last byte of data was stored by a stack operation
- When data is to be placed on the stack, the SP increments before storing data on the stack so that the stack grows up as data is stored
- As data is retrieved from the stack, the byte is read from the stack and the SP decrements to point to the next available byte of stored data

8051 – Stack Pointer example



- SP is set to 07h when the 8051 is reset and can be changed by the programmer
- The stack is limited in height to the size of the internal RAM. If the programmer is not careful when using the stack, the stack can overwrite valuable data in the registere banks, bit-addressable RAM, and other RAM areas
- As a recommendation, place the stack in high addresses of the RAM

8051 – Internal ROM

- Internal ROM occupies code address space 0000h to 0FFFh.
- PC is used to address program code bytes from 0000h to FFFFh
- Program addresses higher than 0FFFh cause the 8051 to fetch code bytes from external memory
- Code can also be fetched exclusively from an external memory by connecting the external access pin (EA') to ground

8051 – Addressing Modes

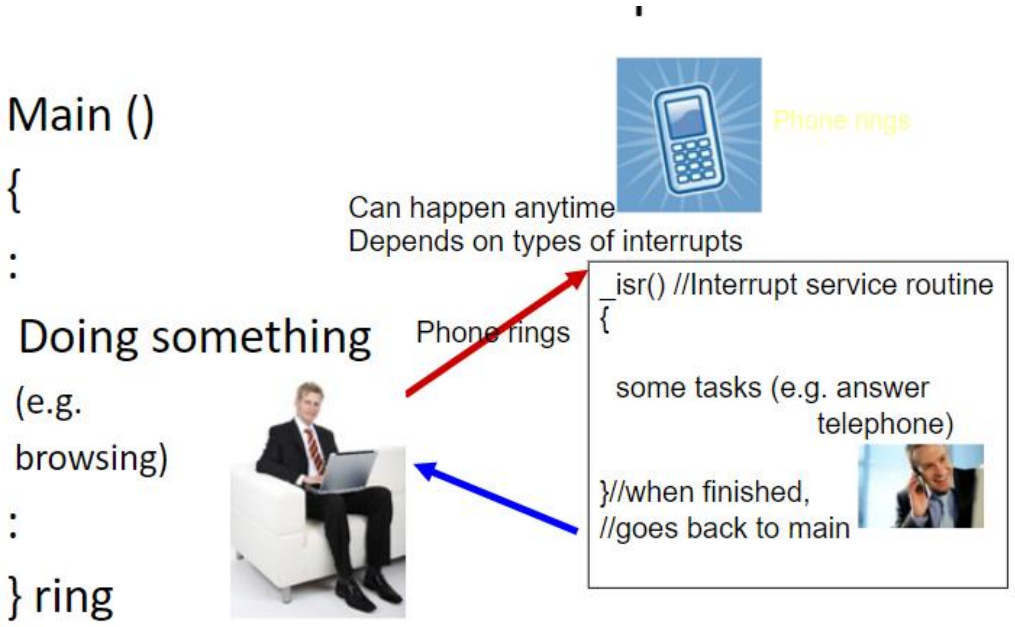
- Five types of addressing modes:
  - **Register addressing:** Operand is placed in one of the general-purpose registers. Data is copied from one register to another
    - Example: ADD A, R0
  - **Direct addressing:** Data is directly copied from the given address to the register
    - Example: ADD A, 30h; (add the contents of register A to the contents of the register whose address is 30h)
  - **Register indirect addressing:** The contents of R0 or R1 is used as the address pointer to locate in a 256 byte block
    - Example: ADD A, @R0
  - **Immediate addressing:** The data is specified in the instruction itself
    - Example: ADD A, #10;
  - **Index addressing:** DPTR is used as a base address, and the Acc carries a constant to which the DPTR is summed to form the memory address
    - Example: MOVC A, @A + DPTR;

Interrupts

- A computer program has two ways to determine the conditions that exist in internal and external circuits
- One method uses software instructions that jump on the states of flags and port pins (polling)
- The second responds to hardware signals, called interrupts, that force the program to call a sub-routine
- An interrupt is an external or internal event that causes the processor to change the normal flow of a program
- Interrupts serve the processor to serve several devices

Interrupts

- Serving several devices can be done in two ways
  - Interrupt
  - Polling



```
Main ()
{
:
 Doing something
(e.g.
browsing)
:
} ring
```

```
_isr() //Interrupt service routine
{

  some tasks (e.g. answer
                  telephone)

}//when finished,
//goes back to main
```

Interrupts

- Interrupt
  - When a device needs to be serviced, it notifies the processor by sending an interrupt signal
  - Upon receiving the interrupt signal, the processor interrupts whatever it is doing and serves the device by executing the program associated to servicing the device
  - The program which is associated to the interrupt is called the ISR – Interrupt Service Routine or Interrupt Handler
- Polling
  - The processor continuously monitors the status of a given device
  - When the condition is met, the processor services the device
  - After that, it moves on to monitor the next device until every device is serviced

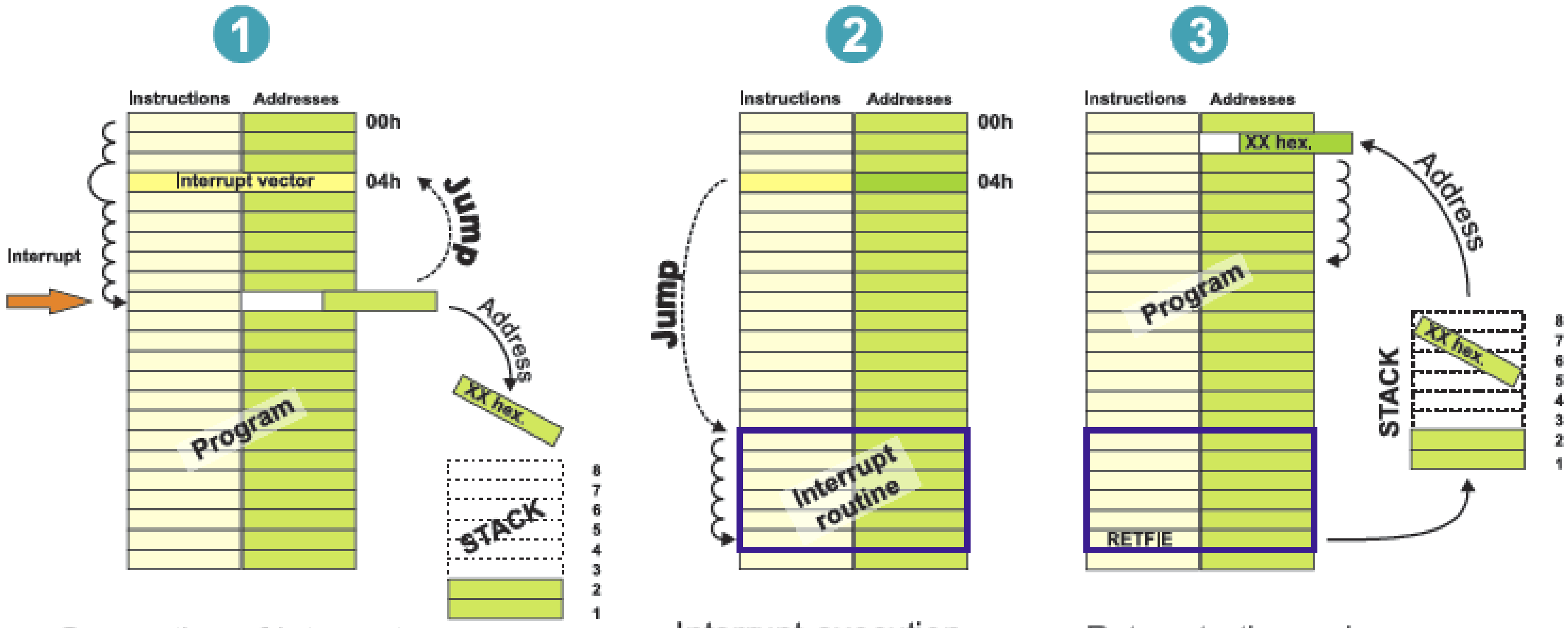
Interrupts

- Polling method is not efficient since it wastes much of the microcontroller;s time by polling devices tht do not need service.
- The advantage of interrupts is that the processor can serve many devices.
- Each device gets the attention of the processor based on the assigned priority
- For polling there is no priority since all devices are checked in a round-robin fashion

Interrupts

- How do interrutps work
  - The device notifies the processor by sending an interrupt signal
  - The processor interrupts whatever it is doing and saves the address of the next instruction (PC) on the stack pointer (SP)
  - The processor jumps to a fixed location in memory, called the interrupt vector table, that holds the address of the ISR. Each interrupt has its own ISR
  - The processor gets the address of the ISR and jumps to it
  - The processor executes the ISR until the last instruction is reached which is a RETI (return from interrupt) instruction
  - Upon reaching RETI, the processor returns to the place where it was interrupted
    - It gets the PC address from the stack pointer
    - It resumes execution from that address

Interrupts



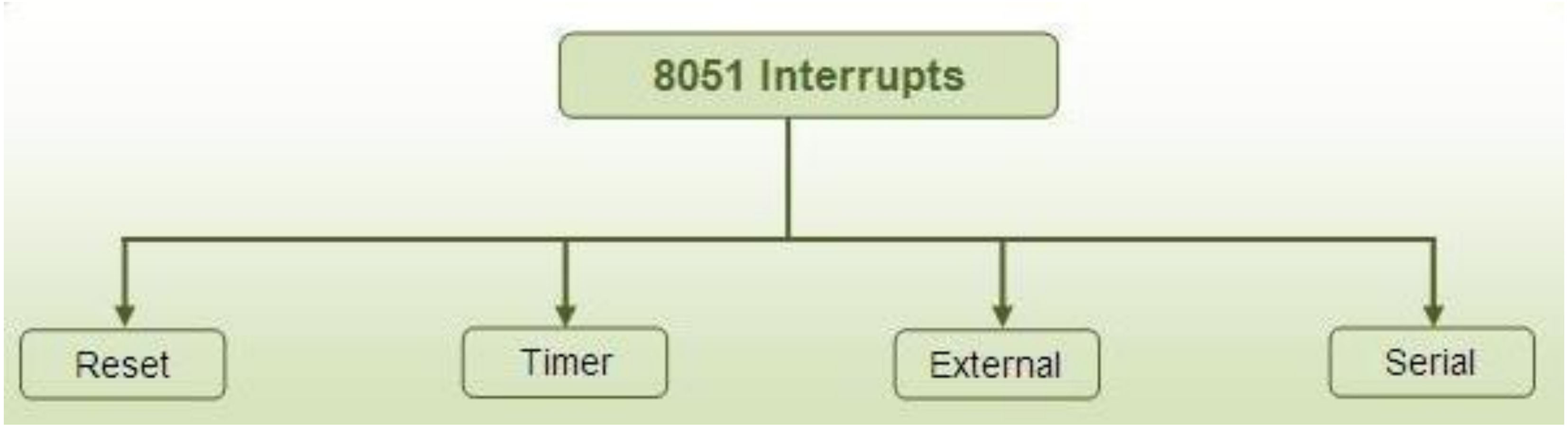
Generation of interrupt

Interrupt execution

Return to the main program

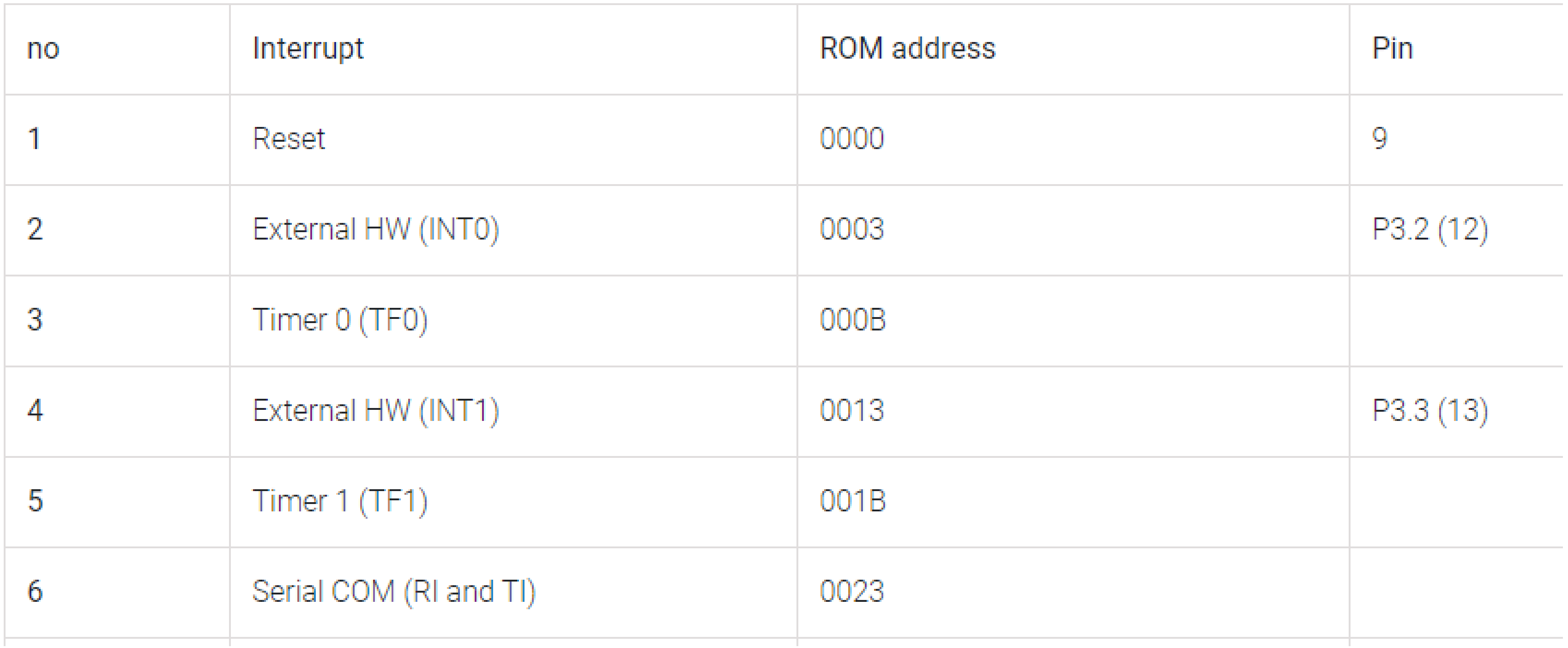
8051 - Interrupts

- 8051 has 6 interrupt sources
  - Reset
  - Two external (INT0, INT1)
  - Two timer interrupts (TF0, TF1)
  - Serial interrupt for serial comms



8051 - Interrupts

- **Interrupt Vector Table**
- Upon reset, all interrupts are disabled (masked), i.e., none will be responded to by the processor if they are activated



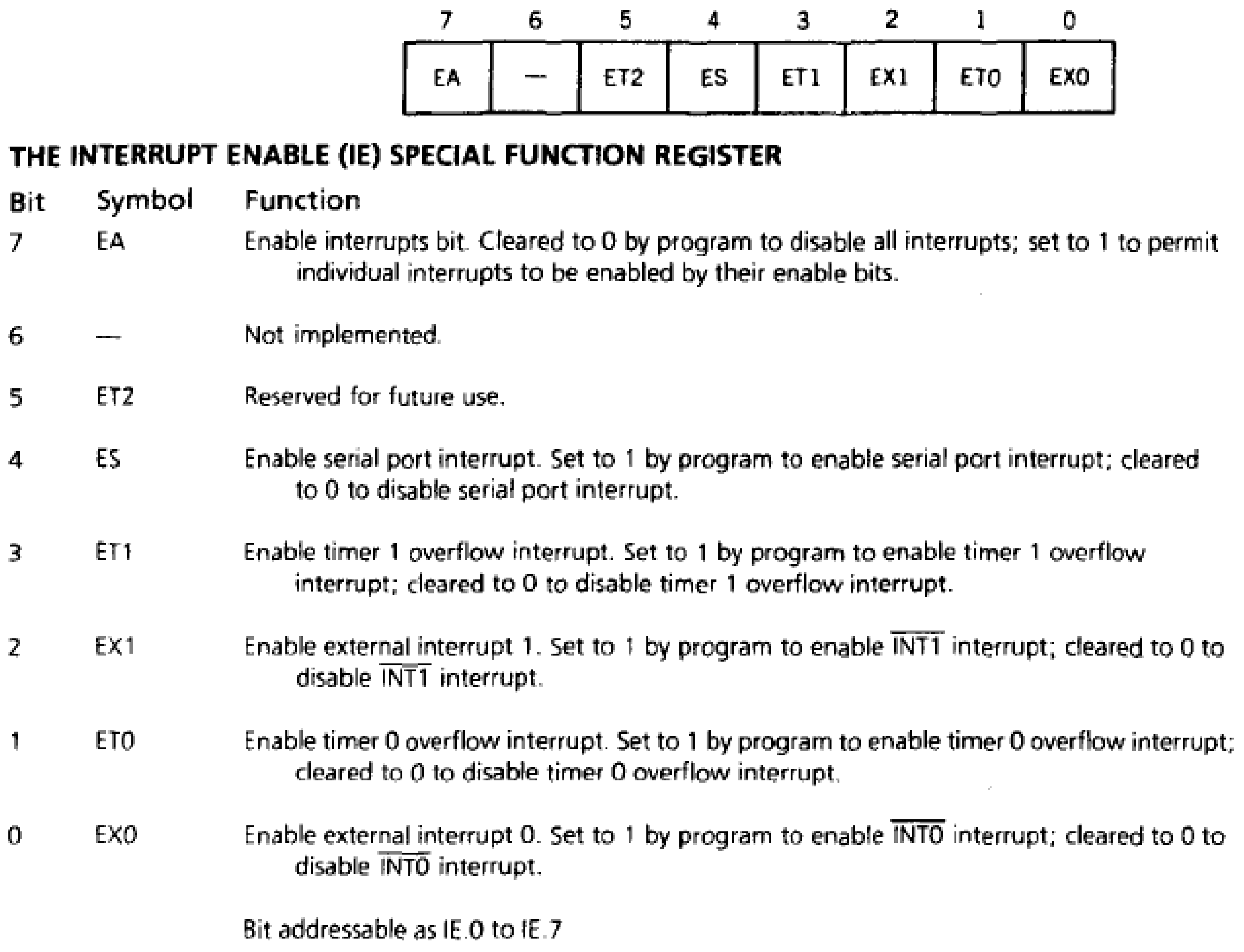
no	Interrupt	ROM address	Pin
1	Reset	0000	9
2	External HW (INT0)	0003	P3.2 (12)
3	Timer 0 (TF0)	000B	
4	External HW (INT1)	0013	P3.3 (13)
5	Timer 1 (TF1)	001B	
6	Serial COM (RI and TI)	0023	

8051 - Interrupts

- Two registers are used for Interrupts
  - Interrupt Enable Register (IE)
  - Interrupt Priority Register (IP)
- The programmer can block all or any combination of the interrupts from acting on the program by setting/clearing bits in these registers

8051 - Interrupts

- Interrupt Enable Register (IE)



7	6	5	4	3	2	1	0
EA	—	ET2	ES	ET1	EX1	ET0	EX0

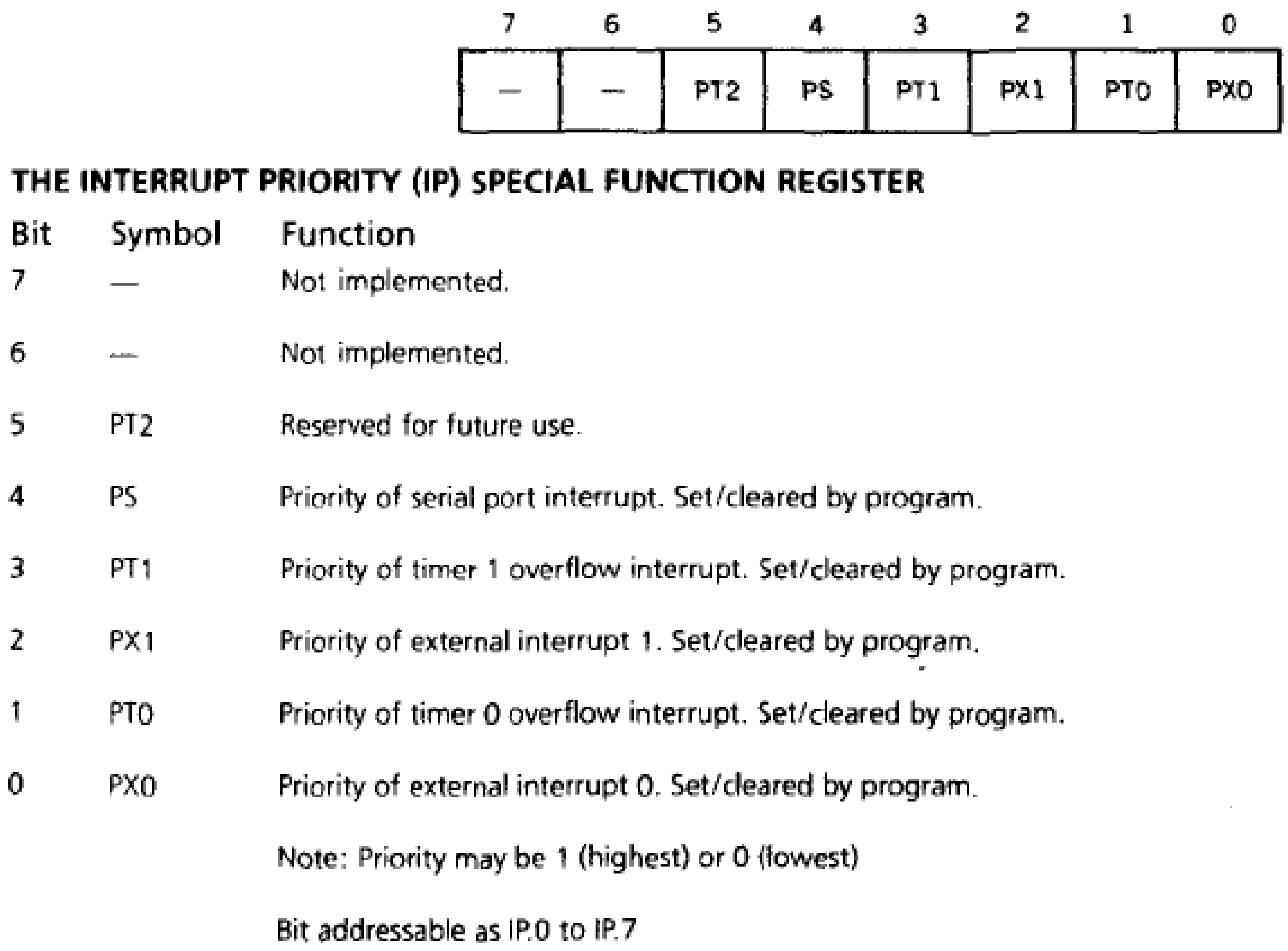
THE INTERRUPT ENABLE (IE) SPECIAL FUNCTION REGISTER

Bit	Symbol	Function
7	EA	Enable interrupts bit. Cleared to 0 by program to disable all interrupts; set to 1 to permit individual interrupts to be enabled by their enable bits.
6	—	Not implemented.
5	ET2	Reserved for future use.
4	ES	Enable serial port interrupt. Set to 1 by program to enable serial port interrupt; cleared to 0 to disable serial port interrupt.
3	ET1	Enable timer 1 overflow interrupt. Set to 1 by program to enable timer 1 overflow interrupt; cleared to 0 to disable timer 1 overflow interrupt.
2	EX1	Enable external interrupt 1. Set to 1 by program to enable $\overline{INT1}$ interrupt; cleared to 0 to disable $\overline{INT1}$ interrupt.
1	ET0	Enable timer 0 overflow interrupt. Set to 1 by program to enable timer 0 overflow interrupt; cleared to 0 to disable timer 0 overflow interrupt.
0	EX0	Enable external interrupt 0. Set to 1 by program to enable $\overline{INT0}$ interrupt; cleared to 0 to disable $\overline{INT0}$ interrupt.

Bit addressable as IE.0 to IE.7

8051 - Interrupts

- Interrup Priority Register (IE)



7	6	5	4	3	2	1	0
—	—	PT2	PS	PT1	PX1	PT0	PX0

THE INTERRUPT PRIORITY (IP) SPECIAL FUNCTION REGISTER

Bit	Symbol	Function
7	—	Not implemented.
6	—	Not implemented.
5	PT2	Reserved for future use.
4	PS	Priority of serial port interrupt. Set/cleared by program.
3	PT1	Priority of timer 1 overflow interrupt. Set/cleared by program.
2	PX1	Priority of external interrupt 1. Set/cleared by program.
1	PT0	Priority of timer 0 overflow interrupt. Set/cleared by program.
0	PX0	Priority of external interrupt 0. Set/cleared by program.

Note: Priority may be 1 (highest) or 0 (lowest)

Bit addressable as IP.0 to IP.7

Microprocessor Classification

- Three main categories
  - RISC Processor
  - CISC Processor
  - Special Purpose Processors

ARM®

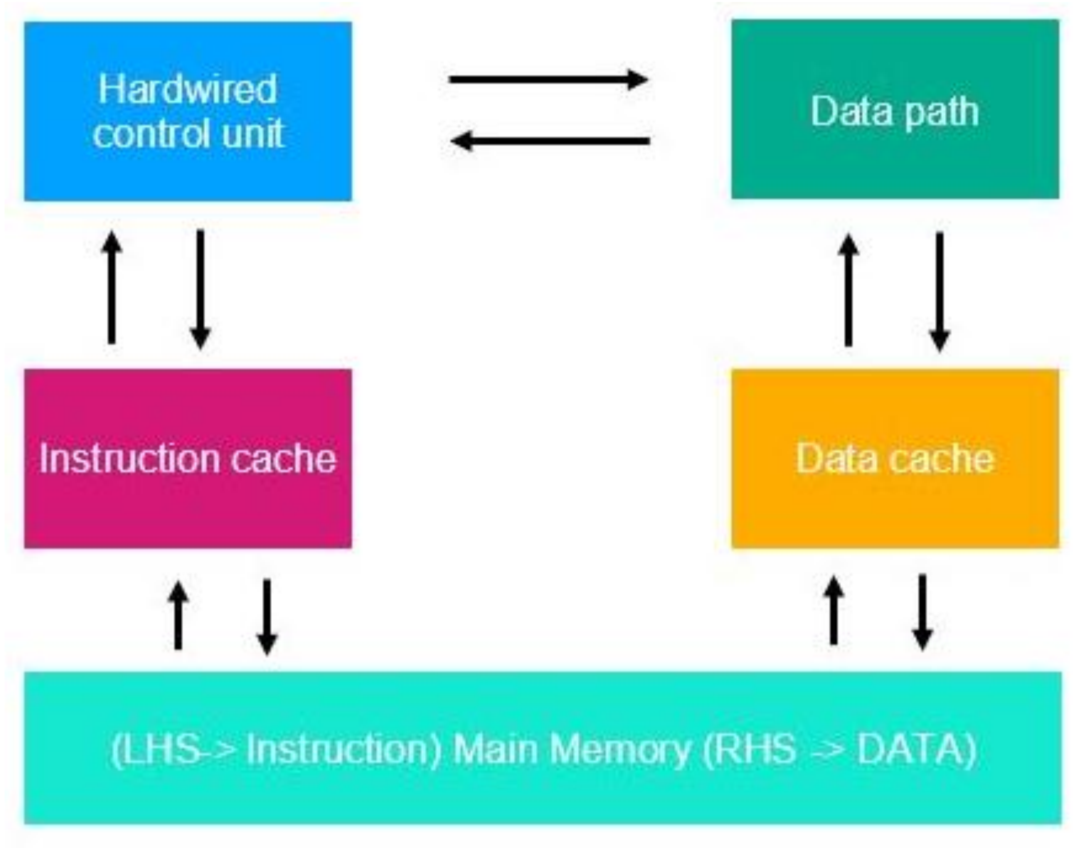
VS

intel®

RISC Processor

- Reduced Instruction Set Computer
- Designed to reduce the execution time by simplifying the instruction set of the computer
- Each instruction requires only one cycle to execute results in uniform execution time. Instruction set is highly optimized
- This reduces efficiency as there ara more lines of code and hence more RAM is needed to store instructions
- The compiler has to work more in converting instructions into machine code
- Some RISC processors
  - ARM
  - MIPS
  - Power PC
  - Etc.

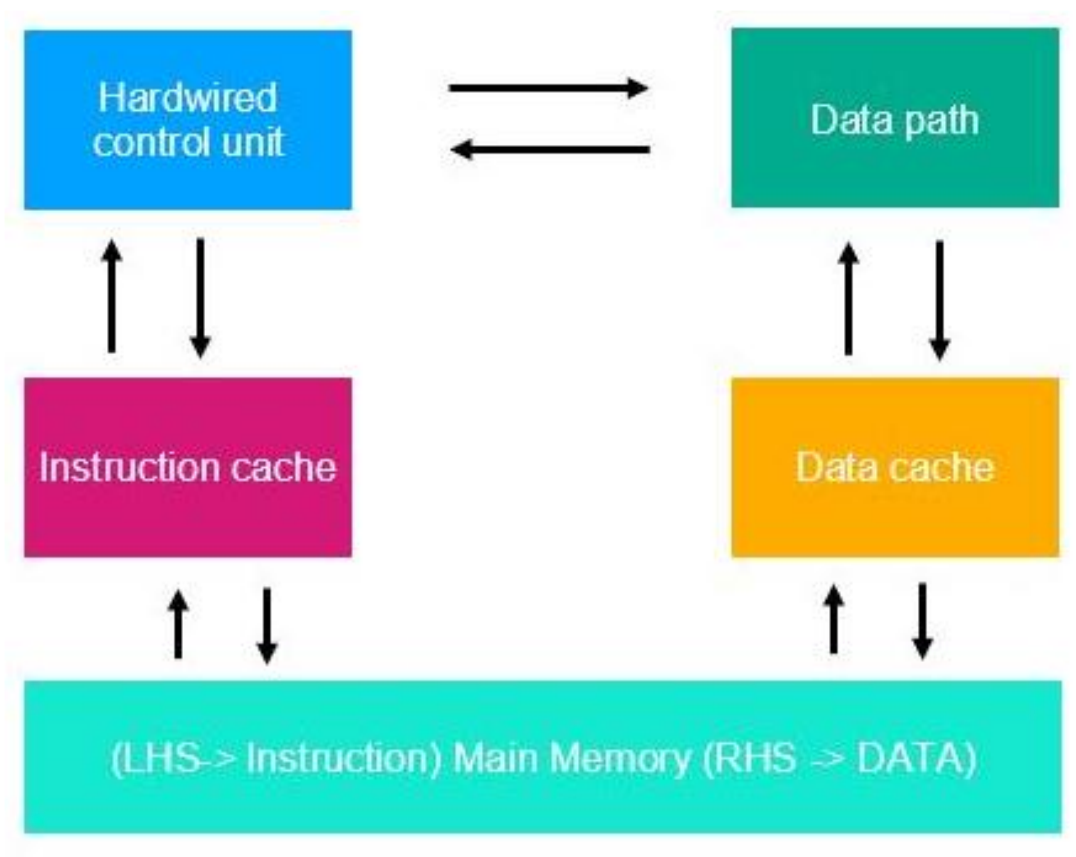
RISC Processor Features



Main memory holds instruction sets and data sets

- Harvard architecture
- Simple instructions
- Supports various data types formats
- Simple addressing modes and fixed length instructions for pipelining
- Registers can be used in any context
- One cycle execution time
- LOAD and STORE instructions are used to access memory locations

RISC Processor Features



Main memory holds instruction sets and data sets

- RISC prevents various interactions with memory by having a large number of registers
- Pipelining is relatively simple cause instructions are done in a uniform interval of time
- Three separate instructions are processed at the same time (fetch, decode, execute)
- More RAM is required

Advantages of RISC processors

- RISC processors make use of registers to pass arguments and to hold local variables
- Fixed-length instructions are easy to pipeline
- Execution time is minimized
- Simple architecture that permits low power consumption

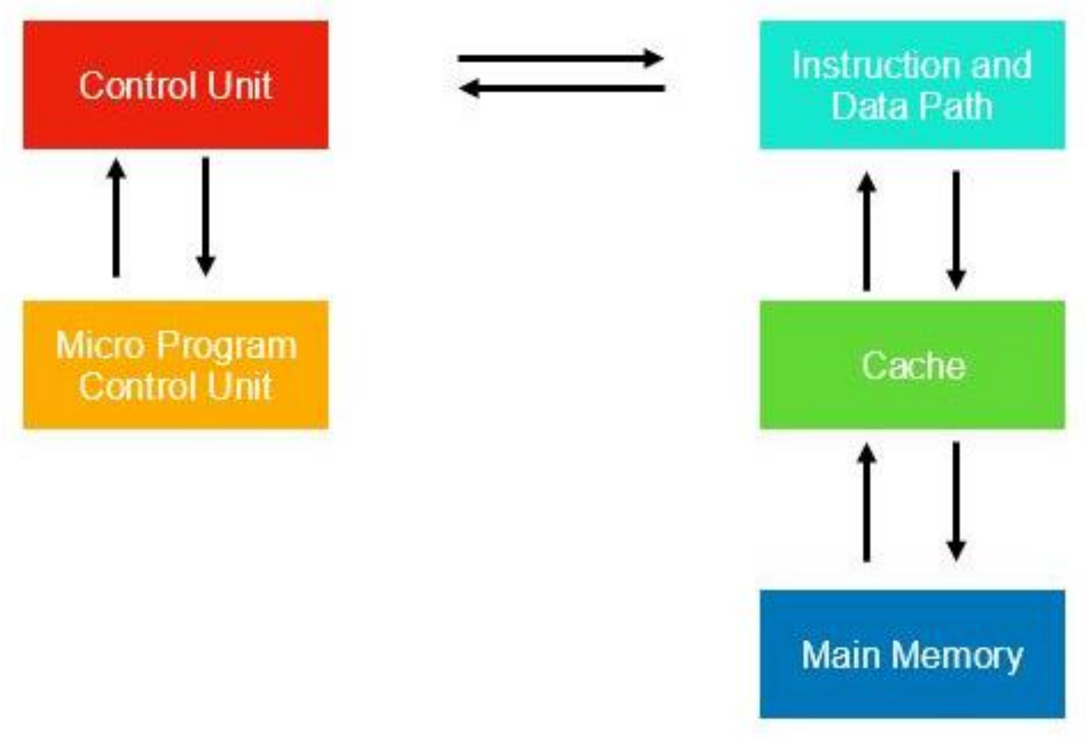
Disadvantages of RISC processors

- Performance strongly depends on the compiler and programmer skills
- RISC processor require very fast memory systems to feed various instructions
- Large cache memory is required

CISC Processor

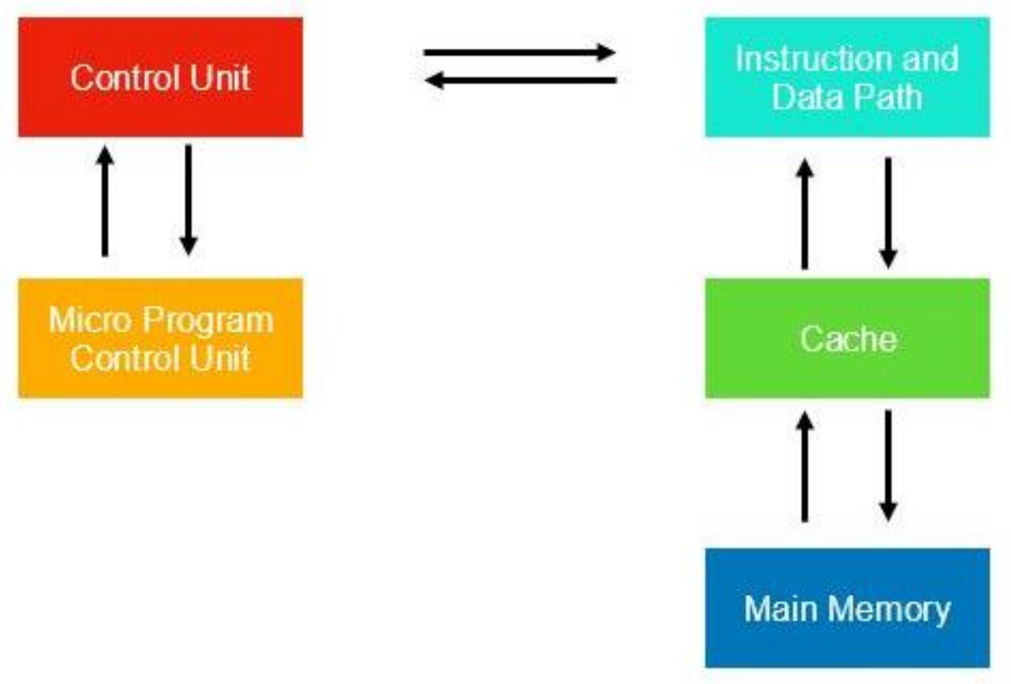
- Complex Instruction Set Computer
- Single instructions can execute several low-level operations (for instance, "load from memory an arithmetic operation and a memory store")
- The number of cycles per instructions vary from instruction to instruction
- Compiler has to do very little work cause the length of the code is relatively short
- Small RAM is required to store instructions
- Capable of executing multi-step operations or addressing modes with single instructions
- CISC processor examples:
  - IBM 370/168
  - Intel 80486
  - High performance custom computers

CISC Processor Features



- To minimize program size, the number of instructions in a program is decreased
- To do this, CISC has to embed some of the low level instructions in single complex instruction
- When it is decoded, the "big" instruction generates several microinstructions to execute

CISC Processor Features



- **Microprogram Control Unit** – uses a seres of microinstructions of the microprogram stored in the "control memory" of the microprogram control unit and generates control signals
- **Control Unit** – accesses the control signals to operate the processor hardware
- **Instructions and data path** – retrieve/fetch opcode and operands of the instructions from the memory
- **Cache and Main Memory** – location where program instructions and operands reside

CISC Processor Features

- LOAD, STORE and MOVE instructions are used to operate with memory
- Variety of addressing modes
- Larger number of instructions
- Variable length of instruction formats
- Several cycles may be required to execute one instruction
- Instruction decoding logic is complex. Example:
    - Auto-increment mode
    - Auto-decrement mode
    - Relative mode

Advantages of CISC Processors

- Memory requirements are minimized due to code size
- Memory access is more flexible due to the complex addressing mode
- Memory locations can be directly accessed by CISC instructions

Disadvantages of CISC Processors

- Code requires several clock cycles to execute a single instruction
- Implementation of pipelining is complicated
- In order to simplify the software, the hardware structure has to be more complicated
- A little bit of history:
  - CISC was designed to minimise the memory requirement when memory was smaller and more expensive. However nowadays memory is inexpensive, and the majority of new computer systems have a large amount of memory, compared to the 1970's when CISC first emerged.

Special Processors

- Designed for special purposes. Examples:
  - Coprocessor
    - Math coprocessor
    - Input/Output processor
      - DMA controllers
      - Graphic display controllers
    - Bus controllers
  - DSPs
    - Obviously, for signal processing applications such as audio/video/image processing, 2D/3D graphics, etc.

Conclusions

- An embedded system is comprised of a digital computer and peripherals (I/O)
- Harvard architectures are the preferable choice in embedded systems
- RISC processors are advantageous over CISC is embedded applications