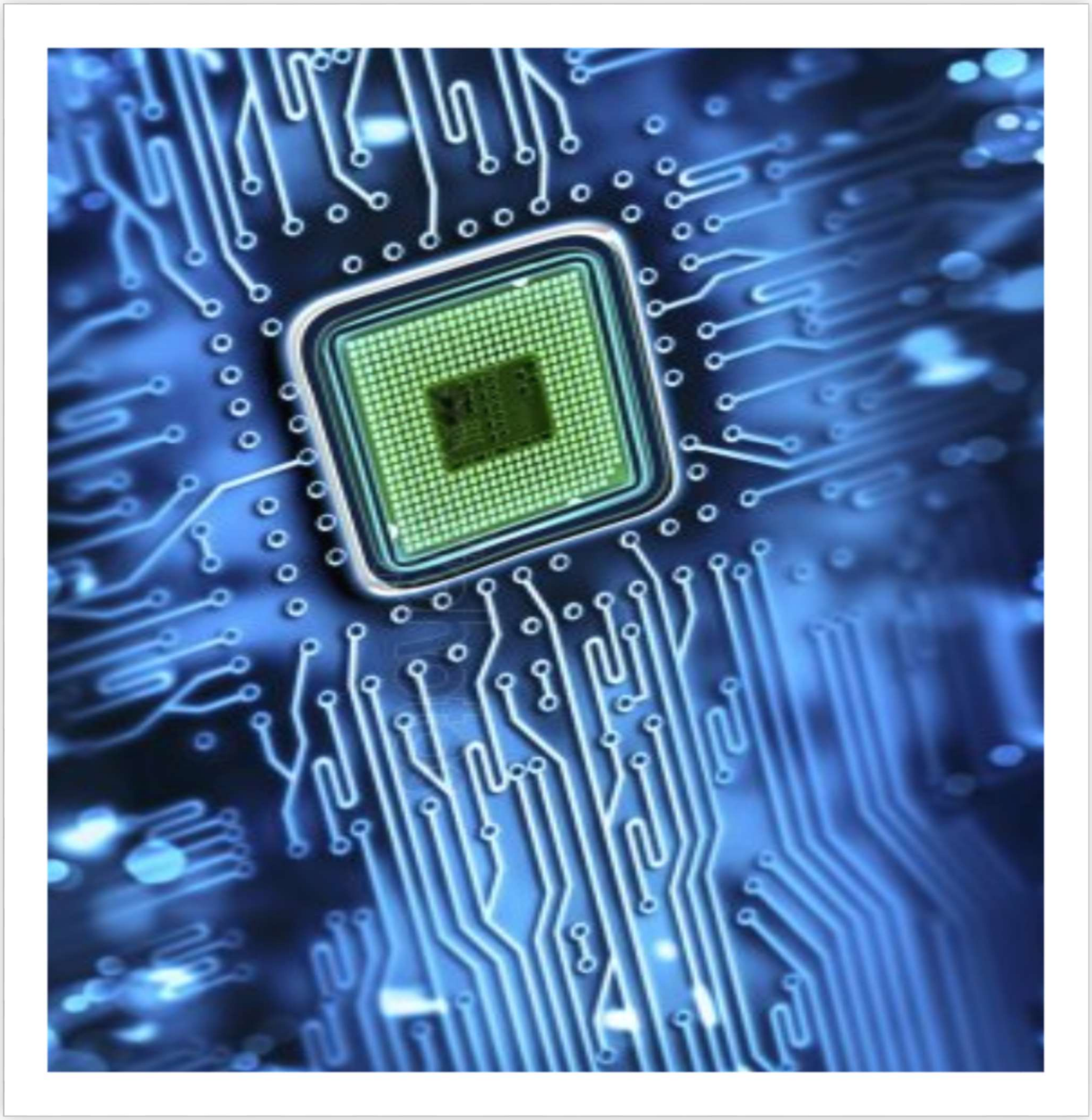


Microprocessors and Assembly Language

CSE 3109 BY BP

Microcomputer System

Introduction to different types of microprocessors and its applications

Generally, the microprocessor is an integrated circuit and it incorporates core function of a computer's central processing unit. The microprocessor is a programmable multipurpose silicon chip, register based, clock driven, it accepts input as a binary data and after the processing, it provides the output data as per the instructions stored in the memory.

- **First Generation Microprocessors**
  The first generation microprocessors were introduced in the year 1971-1972. The instructions of these microprocessors were processed serially, they fetched the instruction, decoded and then executed it. When an instruction of the microprocessor was finished, then the microprocessor updates the instruction pointer & fetched the following instruction, performing this consecutive operation for each instruction in turn.
- **Second Generation Microprocessors**
  In the year 1970, small amount of transistors was available on the integrated circuit in the second generation microprocessors. Examples of the second generation microprocessors are 16-bit arithmetic 7 pipelined instruction processing, MC68000 Motorola microprocessor. These processors are introduced in the year 1979, and Intel 8080 processor is another example of the microprocessor. The second generation of the microprocessor is defined by overlapped fetch, decode and execute the steps. When the first instruction is processed in the execution unit, then the second instruction is decoded and the third instruction is fetched.
  The difference between the first generation microprocessor and second generation microprocessors was mainly the use of new semiconductor technologies to manufacture the chips. The result of this technology resulted in a fivefold increase in instruction, speed, execution and higher chip densities.
- **Third Generation Microprocessors**
  The third generation microprocessors were introduced in the year 1978, as denoted by Intel's 8086 and the Zilog Z8000. These were 16-bit processors with a performance like mini computers. These types of microprocessors were different from the previous generations of microprocessors in that all main workstation industrialists began evolving their own ISC based microprocessor architectures.
- **Fourth Generation Microprocessors**
  As many industries converted from commercial microprocessors to in house designs, the fourth generation microprocessors are entered with outstanding design with a million transistors. Leading edge microprocessors like Motorola's 88100 and Intel's 80960CA could issue & retire more than one instruction per clock cycle.
- **Fifth Generation Microprocessors**
  Fifth generation microprocessors employed decoupled super scalar processing, and their design soon exceeded 10 million transistors. In fifth generation, PCs are a low-margin, high volume business conquered by a single microprocessor.

Organization of Intel 8086/8088 Microprocessor

Read from Assembly Language Programming and Organization of the IBM PC by Marut

- 1.1: The Components of a microcomputer system
  - 1.1.2: The CPU
- 1.2: Instruction Execution
- 1.4: Programming Languages
  - Assembly language
  - Advantages of high level languages

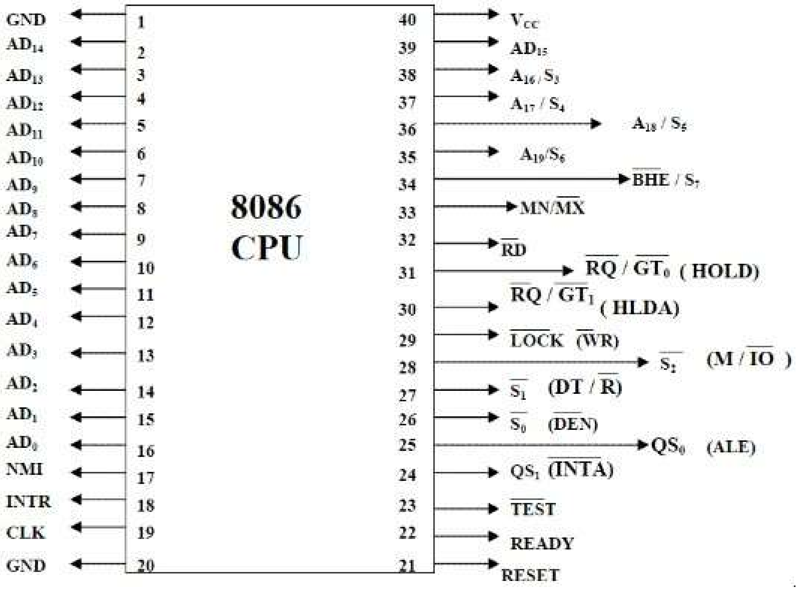
- Advantages of Assembly language

- Address bus: the set of electrical pathways for address signals
- Bus: a set of wires or connections connecting the CPU, memory and I/O ports.
- Bus Interface Unit (BIU): part of the CPU that facilitates communication between the CPU, memory and I/O ports.
- Control bus: the set of electrical paths for control signals.
- Data bus: the set of electrical paths for data signals.
- Opcode: numeric or symbolic code denoting the type of operation for an instruction.
- 2.4: How Integers Are Represented in the Computer
  - 2.4.1: Unsigned Integers
  - 2.4.2: Signed Integers
- 2.5: Character Representation
  - ASCII Code
  - The Keyboard

Features of 8086 Microprocessor

The most important features of the microprocessor are following:

1. To improve the performance of this microprocessor there are two stages of pipelining, which are fetching & execute stage.
2. The fetch stage can transfer the data in 6 bytes of instructions and stored in a line.
3. The execute stage will execute the instructions.
4. The 8086 microprocessor consists of 2900 transistors and it has 256 vectored interrupts.
5. It is the first 16-bit processor with 16 bit ALU & register, internal data bus and 16-bit external data bus.



We can access the execution unit and flag using code. We cannot access the temporary unit using code. The microprocessor uses it itself.

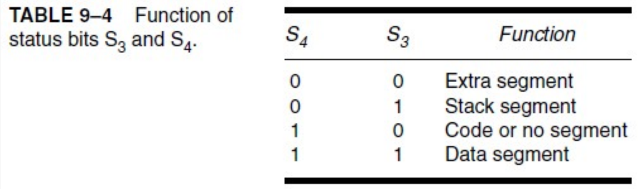
Read from Assembly Language Programming and Organization of the IBM PC by Marut

- 3.1: The Intel 8086 Family of Microprocessors
- 3.2: Organization of the 8086/8088 Microprocessors
  - 3.2.1: Registers
  - 3.2.2: Data Registers
  - 3.2.3: Segment Registers

- 3.2.4: Pointer and Index Registers
- 3.2.5: Instruction Pointer
- 3.2.6: Flags Register

- 3.3: Organization of the PC
  - 3.3.4: Start-up Operation (optional)
- Boot program: the routine that loads the operating system during start-up.
- COMMAND.COM: the command processor for DOS.
- Paragraph: 16 bytes
- Protected mode (virtual address): A processor mode in which the memory used by one program is protected from the actions of another program.
- Real address mode: A processor mode in which the addresses used in a program correspond to a physical memory address.

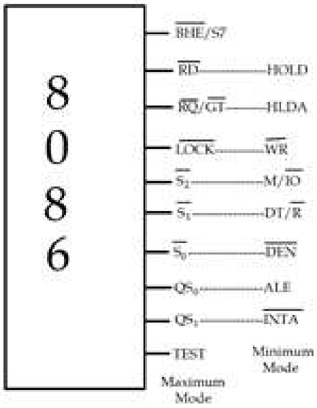
8086 Pin Connections

<table>
<tr><td>$\overline{RD}$</td><td>Whenever the read signal is a logic 0, the data bus is receptive to data from the memory or I/O devices connected to the system. This pin floats to its high-impedance state during a hold acknowledge.</td></tr>
<tr><td>$AD_0 - AD_{15}$</td><td>16 pins represents 16bits of address bus</td></tr>
<tr><td>$A_{16} - A_{19}$</td><td>The address/status bus bits are multiplexed to provide address signals $A_{16} - A_{19}$ and also status bits $S_3 - S_6$. These pins also attain a high-impedance state during the hold acknowledge.
So, total 20 pins (4 here and 16 above) represents 20bit address bus.</td></tr>
<tr><td>$\overline{BHE}$</td><td>The <b>bus high enable</b> pin is used in the 8086 to enable the most-significant data bus bits ($D_8 - D_{15}$) during a read or a write operation. The state of S_7 is always a logic 1.</td></tr>
<tr><td>$D_0 - D_{15}$</td><td></td></tr>
<tr><td>$D_8 - D_{15}$</td><td>8 bit for bus</td></tr>
<tr><td>$MN/\overline{MX}$</td><td>The <b>minimum/maximum</b> mode pin selects either minimum mode or maximum mode operation for the microprocessor. If minimum mode is selected, the $MN/\overline{MX}$ pin must be connected directly to +5.0 V. When on minimum mode, microprocessor only operates itself.</td></tr>
<tr><td>S_3, S_4</td><td>
TABLE 9–4 Function of status bits S_3 and S_4.
<table>
<tr><th>S_4</th><th>S_3</th><th>Function</th></tr>
<tr><td>0</td><td>0</td><td>Extra segment</td></tr>
<tr><td>0</td><td>1</td><td>Stack segment</td></tr>
<tr><td>1</td><td>0</td><td>Code or no segment</td></tr>
<tr><td>1</td><td>1</td><td>Data segment</td></tr>
</table></td></tr>
<tr><td>S_5</td><td>Accept request from INTR</td></tr>
<tr><td>S_6</td><td>Internal bus control 8288</td></tr>
<tr><td>READY</td><td>The <b>READY</b> input is controlled to insert wait states into the timing of the microprocessor. If the READY pin is placed at a logic 0 level, the microprocessor enters into wait states and remains idle. If the READY pin is placed at a logic 1 level, it has no effect on the operation of the microprocessor.</td></tr>
<tr><td>RESET</td><td>The <b>reset</b> input causes the microprocessor to reset itself if this pin is held high for a minimum of four clocking periods. Whenever the 8086 or 8088 is reset, it begins executing instructions at memory location FFFF0h (IP is 0000h and CS is FFFFh) and disables future interrupts by clearing the IF flag bit (all flags are cleared). Instruction queue is cleared.</td></tr>
</table>

8086 addressing models

- Read from Intel Microprocessors: Architecture, Programming and Interfacing by Barry B. Brey
- 9.1: PIN-OUTS AND THE PIN FUNCTIONS

- The figure (from book) contains the pin diagram of 8086 only, not 8088.



- The pin-out
- Power supply requirements (optional)
- DC characteristics (optional)
- Pin connections
  i. Minimum mode pins
  ii. Maximum mode pins

Minimum Mode

$DT/\overline{R}$: The **data transmit/receive** signal shows that the microprocessor data bus is transmitting ($DT/\overline{R} = 1$) or receiving ($DT/\overline{R} = 0$) data. This signal is used to enable external data bus buffers.

$M/\overline{IO}$: Address of M/IO. This pin selects memory or I/O. This pin indicates that the microprocessor address bus contains either a memory address or an I/O port address. This pin is at its high-impedance state during a hold acknowledge.

$\overline{WR}$: Data is being written to M/IO. During the time that the pin is a logic 0, the data bus contains valid data for memory or I/O. This pin floats to a high impedance during a hold acknowledge.

$\overline{INTA}/\overline{INTR}$: The **interrupt acknowledge** signal is a response to the $INTR$ input pin. The $\overline{INTA}$ pin is normally used to gate the interrupt vector number onto the data bus in response to an interrupt request.

ALE: **Address latch enable** shows that the 8086/8088 address/data bus contains address information. This address can be a memory address or an I/O port number. Note that the ALE signal does not float during a hold acknowledge.

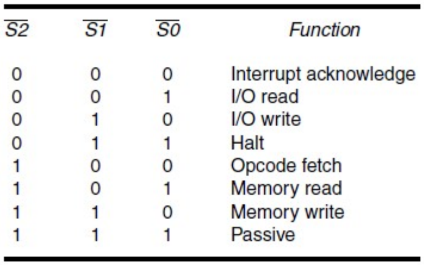
DEN: **Data bus enable** activates external data bus buffers.

$\left.\begin{matrix}HOLD\\HLDA\end{matrix}\right\}DMA$

HOLD: The **hold input** requests a direct memory access (DMA). If the HOLD signal is a logic 1, the microprocessor stops executing software and places its address, data and control bus at the high-

Maximum Mode

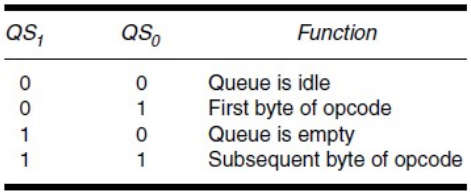
$\overline{S_0}, \overline{S_1}, \overline{S_2}$: The **status bits** indicate the function of the current bus cycle. These signals are normally decoded by the 8288 bus controller. The table below shows the function of these three status bits in the maximum mode.



$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	*Function*
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive

$\overline{RQ}/\overline{GT}$: The **request/grant** pin request direct memory accesses (DMA) during maximum mode operation. This line is bidirectional and is used to both request and grant a DMA operation.

QS_0, QS_1: The **queue status** bits show the status of the internal instruction queue. These pins are provided for access by the numeric coprocessor (8087).



QS_1	QS_0	*Function*
0	0	Queue is idle
0	1	First byte of opcode
1	0	Queue is empty
1	1	Subsequent byte of opcode

$\overline{LOCK}$: The **lock** output is used to lock peripherals off the system. This pin is activated by using the LOCK: prefix on any instruction.

impedance state. If the HOLD pin is a logic 0, the microprocessor executes software normally. HLDA: **Hold acknowledge** indicates that the 8086/8088 has entered the hold state. $\overline{SS_0}$: The $\overline{SS_0}$ status line is equivalent to the S_0 pin in maximum mode operation of the microprocessor. This signal is combined with $IO/\overline{M}$ and $DT/\overline{R}$ to decode the function of the current bus cycle.	

- 9.6: MINIMUM MODE VERSUS MAXIMUM MODE
  - Minimum mode operation
  - Maximum mode operation
- Timing diagram of 8086 BUS cycles
  - 9.4: BUS TIMING (I don't think you'll ever want to read this piece of shit in your fucking miserable life)
    - Read Timing
    - Write Timing

To have some more FUN:

- https://www.geeksforgeeks.org/pin-diagram-8086-microprocessor/
- https://www.slideshare.net/sushantsyadav/8086-modes-54659722

The component of microprocessor system

- **Arithmetic and Logic Unit**
  The "arithmetic and logic unit" (ALU) performs math computations, such as subtraction, addition, division and Boolean functions. Boolean functions are a type of logic used for circuit designs. The ALU also executes comparisons and logic testing. The processor transmits signals to the ALU, which interprets the instructions and performs the calculations.
- **Registers**
  Microprocessors have temporary data holding places called registers. These memory areas maintain data, such as computer instructions, storage addresses, characters and other data. Some computer instructions may require the use of certain registers as part of a command. Each register has a specific function, such as instruction register, program counter, accumulator and memory address register. For example, a program register holds the address of instructions taken from random access memory. (*Remember Instruction Pointer?*)
- **Control Unit**
  Control units (CUs) receive signals from the CPU, which instructs the control unit to move data from microprocessor to microprocessor. The control unit also directs the arithmetic and logic unit. Control units consist of multiple components, such as decoder, clock and control logic circuits. Working together, these devices transmit signals to certain locations on the microprocessor.
  For example, the decoder receives commands from an application. The decoder interprets the instructions and takes an action. It sends signals to the ALU or directs registers to perform specific tasks. The control logic unit transmits signals to different sections of the microprocessor and registers, which informs these components to execute actions. The clock sends signals that synchronize and ensure timely execution of commands and processes.
- **Buses**
  Microprocessors have a system of buses, which move data. Buses refer to classifications of wiring that have specific tasks and functions. The data bus transfers data between the central processing unit and random access memory (RAM) -- the computer's primary memory. The control bus sends information necessary to coordinate and control multiple tasks. The address bus transmits the address between the CPU and the RAM for data being processed.
- **Cache Memory**

Some advanced microprocessors have memory caches, which retain the last data used by the CPU. Memory caches speed up the computing process, because the CPU does not have to go to the slower RAM to retrieve data. Many computers have level 1 or level 2 caches; some systems have level 3 caches. The cache level indicates the order in which the CPU checks for data, starting with level 1. Manufacturers often integrate level 2 and level 3 caches into the microprocessor, which enhances processing speed.

I/O device

- Not important according to Biprodip Pal Sir (as far as I can remember). But you know which are I/O devices. If not, then what the fuck are you doing with your life!

Interrupt structures

- An interrupt is the method of processing the microprocessor by peripheral device. An interrupt is used to cause a temporary halt in the execution of program. Microprocessor responds to the interrupt with an interrupt service routine, which is short program or subroutine that instructs the microprocessor on how to handle the interrupt.
- Read from Intel Microprocessors: Architecture, Programming and Interfacing by Barry B. Brey
- 12.1: BASIC INTERRUPT PROCESSING
  - The purpose of interrupts
  - Interrupts

    There are three types of interrupts

    i. Hardware interrupt: Through INTR (interrupt request) pin. এর মাধ্যমে microprocessor interrupt request নেয়। Microprocessor এই request নিতেও পারে, নাও নিতে পারে (delay করতে পারে). I/O device এর input গুলো এর মাধ্যমে track করে।

    The primary sources of interrupts, however, are the PCs timer chip, keyboard, serial ports, parallel ports, disk drives, CMOS real-time clock, mouse, sound cards, and other peripheral devices.

    ii. Software interrupt

    There are instructions in 8086 which cause an interrupt. They are

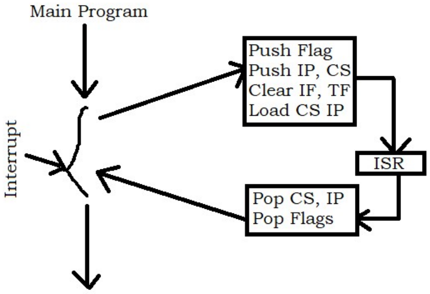
    ➢ INT instructions with type number specified.

    ➢ INT 3, Break Point Interrupt instruction.

    ➢ INTO, Interrupt on overflow instruction.

    These are instructions at the desired places in a program. Software Interrupt instructions can be used to test the working of the various Interrupt handlers.

    iii. Non-maskable interrupt: This occurs when a logic 1 is placed on the NMI input pin to the microprocessor. This input is non-maskable, which means that it cannot be disabled.
  - Operation of a Real Mode Interrupt



This diagram shows what happens after an interrupt occurs. To know more, please read the mentioned article above.

  - Interrupt Flag Bits (optional)
- Interrupt: (*I don't know why did I write these but I did that a long time ago and I have a shitty memory, sorry about that.*)
  - External signal

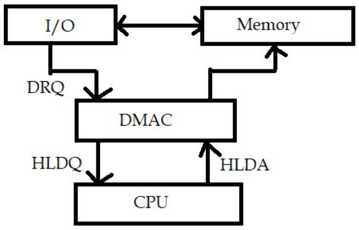
- Special instructions
- Exceptions

I/O interfacing

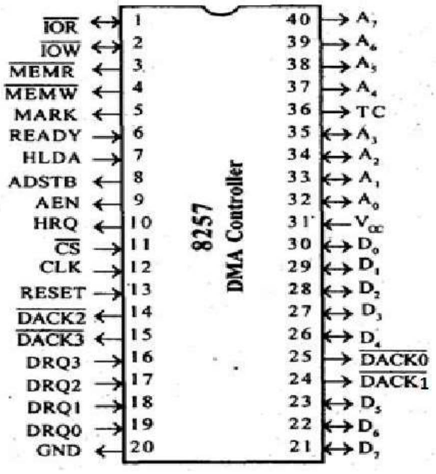
- Not important according to Biprodip Pal Sir.

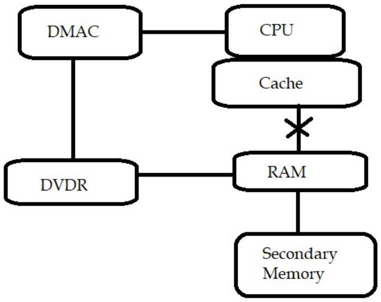
DMA

- **Direct Memory Access** (DMA) is one of several methods for coordinating the timing of data transfers between an input/output (I/O) device and the core processing unit or memory in a computer. DMA is one of the faster types of synchronization mechanisms, generally providing significant improvement over interrupts, in terms of both latency and throughput. An I/O device often operates at a much slower speed than the core. DMA allows the I/O device to access the memory directly, without using the core. DMA can lead to a significant improvement in performance because data movement is one of the most common operations performed in processing applications.
  Traditionally, DMA uses the same internal address and data buses as the core. Consequently, when DMA performs one or more word transfers, it can cause the core to temporarily halt activity for one or more cycles while DMA moves the data. With this type of architecture, the core and DMA cannot both perform data moves in the same core clock cycle.
- Direct Memory Access (DMA) is a method of allowing data to be moved from one location to another in a computer without intervention from the central processor (CPU). It is also a fast way of transferring data within (and sometimes between) computer. The DMA I/O technique provides direct access to the memory while the microprocessor is temporarily disabled. The DMA controller temporarily borrows the address bus, data bus and control bus from the microprocessor and transfers the data directly from the external devices to a series of memory locations (and vice versa).
- Read from Intel Microprocessors: Architecture, Programming and Interfacing by Barry B. Brey
- 13.1: BASIC DMA OPERATION (*timing diagram given in the book is optional for you to understand. The diagram went over my head*)



  - HLDQ: Hold Request, a pin of microprocessor
  - HLDA: Hold Acknowledgement, through this microprocessor gives control of System Bus to DMAC. After data transfer is completed, control is returned to microprocessor.
- 13.2: THE 8237 DMA CONTROLLER





- Memory-to-memory Transfer with the 8237 (optional, page-499)

- *(Don't know what are the functions of these things)*
  - Source address
  - Bytes to transfer
  - Starting address of memory
- 2 cycle vs. 1 Cycle

  আগে memory থেকে CPU থেকে memory, এই দুই cycle এ data transfer হত। এখন memory থেকে memory এই এক cycle এ data transfer হয়।
- **Modes of operation:**
  - **Burst mode/block transfer mode:**

    In *burst mode*, an entire block of data is transferred in one contiguous sequence. Once the DMA controller is granted access to the system bus by the CPU, it transfers all bytes of data in the data block before releasing control of the system buses back to the CPU, but renders the CPU inactive for relatively long periods of time. The mode is also called "Block Transfer Mode".
  - **Cycle stealing**

    The *cycle stealing mode* is used in systems in which the CPU should not be disabled for the length of time needed for burst transfer modes. In the cycle stealing mode, the DMA controller obtains access to the system bus the same way as in burst mode, using *BR (Bus Request)* and *BG (Bus Grant)* signals, which are the two signals controlling the interface between the CPU and the DMA controller. However, in cycle stealing mode, after one byte of data transfer, the control of the system bus is reasserted to the CPU via BG. It is then continually requested again via BR, transferring one byte of data per request, until the entire block of data has been transferred. By continually obtaining and releasing the control of the system bus, the DMA controller essentially interleaves instruction and data transfers. The CPU processes an instruction, then the DMA controller transfers one data value, and so on. On the one hand, the data block is not transferred as quickly in cycle stealing mode as in burst mode, but on the other hand the CPU is not idled for as long as in burst mode. Cycle stealing mode is useful for controllers that monitor data in real time
  - **Transparent mode**: when CPU is free

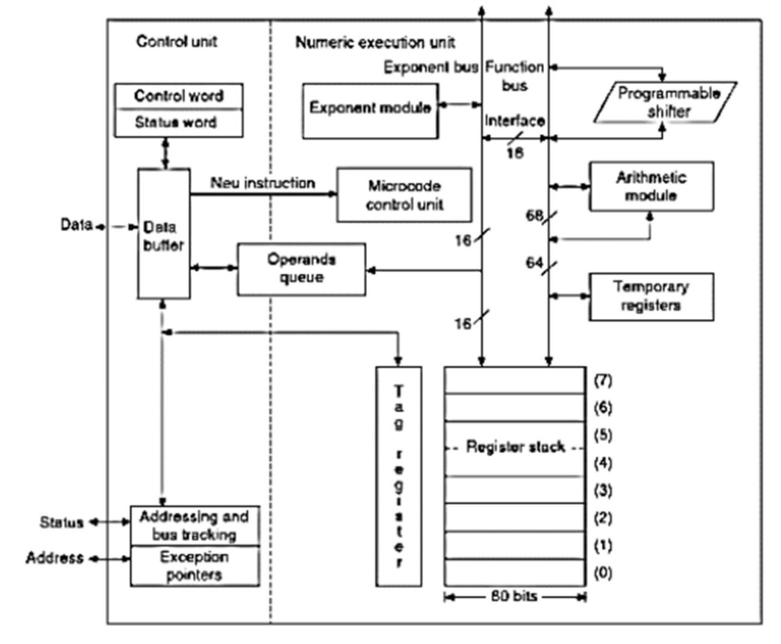
    Transparent mode takes the most time to transfer a block of data, yet it is also the most efficient mode in terms of overall system performance. In transparent mode, the DMA controller transfers data only when the CPU is performing operations that do not use the system buses. The primary advantage of transparent mode is that the CPU never stops executing its programs and the DMA transfer is free in terms of time, while the disadvantage is that the hardware needs to determine when the CPU is not using the system buses, which can be complex. This is also called as "Hidden DMA data transfer mode".
- **Advantages and disadvantages of DMA**

Advantages	**Disadvantages**
- allows a peripheral device to read from/write to memory without going through the CPU. - allows for faster processing since the processor can be working on something else while the peripheral can be populating memory.	- requires a DMA controller to carry out the operation, which increases the cost of the system. - cache coherence problems.

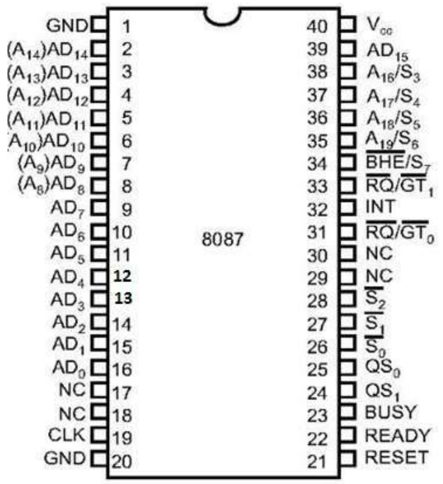
Co-processors

- A coprocessor is a special set of circuits in a microprocessor chip that is designed to manipulate numbers or perform some other specialized function more quickly than the basic microprocessor circuits could perform the same task. A coprocessor offloads specialized processing operations, thereby reducing the burden on the basic microprocessor circuitry and allowing it to work at optimum speed. By offloading processor-intensive tasks from the main processor, coprocessors can accelerate system performance. Coprocessors allow a line of computers to be customized, so that customers who do not need the extra performance do not need to pay for it.

- Used in arithmetic operation.
- **8087**



Block Diagram of 8087 math coprocessor



8087 numeric data processor is also known as **Math co-processor, Numeric processor extension** and **Floating point unit**. It was the first math coprocessor designed by Intel to pair with 8086/8088 resulting in easier and faster calculation. Once the instructions are identified by the 8086/8088 processor, then it is allotted to the 8087 co-processor for further execution.

The data types supported by 8087 are:

- Binary Integers
- Packed decimal numbers
- Real numbers
- Temporary real format

The most prominent **features of 8087** numeric data processor are as follows:

- It supports data of type integer, float and real types ranging from 2-10 bytes.
- The processing speed is so high that it can calculate multiplication of two 64-bits real numbers in ~27 µs and can also calculate square-root in ~35 µs.
- It follows IEEE floating point standards.

RISC processors

- A **reduced instruction set computer**, or RISC, is one whose **instruction set architecture (ISA)** allows it to have fewer **cycles per instruction (CPI).** It is a type of microprocessor that has a limited number of instructions. They can execute their instructions very fast because instructions are very small and simple. RISC chips require fewer transistors which make them cheaper to design and produce. In RISC, the instruction set contains simple and basic instructions from which more complex instruction can be produced. Most instructions complete in one cycle, which allows the processor to handle many instructions at same time. In this, instructions are register based and data transfer takes place from register to register. This technology requires more registers which is why these microprocessors have more general purpose registers.

  Example: ARC, ARM, AVR, Mac, IBM
- **Characteristics:**

1. Simpler instruction, hence simple instruction decoding.
2. Instruction come under size of one word.
3. Instruction take single clock cycle to get executed.
4. More number of general purpose register.
5. Simple Addressing Modes.
6. Less Data types.
7. Pipelining can be achieved.

- **Advantages of RISC processor architecture**
  - Because of the small set of instructions of RISC, high-level language compilers can produce more efficient code.
  - RISC allows freedom of using the space on microprocessors because of its simplicity.
  - Instead of using Stack, many RISC processors use the registers for passing arguments and holding the local variables.
  - RISC functions uses only a few parameters and the RISC processors cannot use the call instructions and therefore, use a fixed length instruction which are easy to pipeline.
  - The speed of the operation can be maximized and the execution time can be minimized.
  - Very less number of instruction formats (less than four), a few number of instructions (around 150) and a few addressing modes (less than four) are needed.
- **Drawbacks of RISC processor architecture**
  - With the increase in length of the instructions, the complexity increases for the RISC processors to execute due to its character cycle per instruction.
  - The performance of the RISC processors depends mostly on the compiler or programmer as the knowledge of the compiler plays a major role while converting the CISC code to a RISC code; hence, the quality of the generated code depends on the compiler.
  - While rescheduling the CISC code to a RISC code, termed as a code expansion, will increase the size. And, the quality of this code expansion will again depend on the compiler, and also on the machine's instruction set.
  - The first level cache of the RISC processors is also a disadvantage of the RISC, in which these processors have large memory caches on the chip itself. For feeding the instructions, they require very fast memory systems.

PowerPC processor

- **PowerPC** (**Performance Optimization With Enhanced RISC – Performance Computing**, sometimes abbreviated as **PPC**) is a reduced instruction set computer (RISC) instruction set architecture (ISA) created by the 1991 Apple–IBM–Motorola alliance, known as *AIM*. PowerPC, as an evolving instruction set, has since 2006 been named Power ISA, while the old name lives on as a trademark for some implementations of Power Architecture-based processors.

  PowerPC is also an evolving instruction set that has been renamed Power ISA since 2006 and lives on as a legacy architecture. PowerPC CPUs have been popularized in part by Apple due to their use in 1994-2006 workstation Macs, before Apple embraced Intel CPUs. However, the PowerPC was first used in IBM RS/6000 workstations, which ran a Unix-based operating system.

  When the PowerPC microprocessor architecture was designed, it was created as an open standard architecture, and other companies were invited to build on top of it. RISC is based on a principle that the simplest computing instructions are the ones most frequently performed. However, CPUs have also been designed with more complicated instruction sets in mind, allowing for simpler and faster operations, and the ability to perform more instructions per clock speed.

CISC processors

- A **complex instruction set computer** is a computer in which single instructions can execute several low-level operations (such as a load from memory, an arithmetic operation, and a memory store) or are capable of multi-step operations or addressing modes within single instructions.

  The CISC approach attempts to minimize the number of instructions per program, sacrificing the number of cycles per instruction. Computers based on the CISC architecture are designed to decrease the memory cost. Because, the large programs need more storage, thus increasing the memory cost and large memory becomes more expensive. To solve these problems, the number of instructions per program can be reduced by embedding the number of operations in a single instruction, thereby making the instructions more complex.

  Example: IBM 370/168, VAX 11/780, Intel x86

$$CPU\ Time = \frac{Seconds}{Program} = \frac{Instructions}{Program} X \frac{Cycles}{Instructions} X \frac{Seconds}{Cycle}$$

- **Characteristics:**
  1. Complex instruction, hence complex instruction decoding.
  2. Instruction are larger than one-word size.
  3. Instruction may take more than single clock cycle to get executed.
  4. Less number of general purpose register as operation get performed in memory itself.
  5. Complex Addressing Modes.
  6. More Data types.
- **Advantages of CISC architecture**
  - Each machine language instruction is grouped into a microcode instruction and executed accordingly, and then are stored inbuilt in the memory of the main processor, termed as microcode implementation.
  - As the microcode memory is faster than the main memory, the microcode instruction set can be implemented without considerable speed reduction over hard wired implementation.
  - Entire new instruction set can be handled by modifying the micro program design.
  - CISC, the number of instructions required to implement a program can be reduced by building rich instruction sets and can also be made to use slow main memory more efficiently.
  - Because of the superset of instructions that consists of all earlier instructions, this makes micro coding easy.
- **Drawbacks of CISC**
  - The amount of clock time taken by different instructions will be different – due to this – the performance of the machine slows down.
  - The instruction set complexity and the chip hardware increases as every new version of the processor consists of a subset of earlier generations.
  - Only 20% of the existing instructions are used in a typical programming event, even though there are many specialized instructions in existence which are not even used frequently.
  - The conditional codes are set by the CISC instructions as a side effect of each instruction which takes time for this setting – and, as the subsequent instruction changes the condition code bits – so, the compiler has to examine the condition code bits before this happens.

EPIC

- **Explicitly parallel instruction computing** (**EPIC**) is a term coined in 1997 by the HP–Intel alliance to describe a computing paradigm that researchers had been investigating since the early 1980s. This paradigm is also called *Independence* architectures. It was the basis for Intel and HP development of the Intel Itanium architecture, and HP later asserted that "EPIC" was merely an old term for the Itanium architecture. EPIC permits microprocessors to execute software instructions in parallel by using the compiler, rather than complex on-die circuitry, to control parallel instruction execution. This was intended to allow simple performance scaling without resorting to higher clock frequencies.

- **Difference between RISC and CISC processor**

RISC	CISC
1. Have simple instructions taking about one clock cycle. The average clock cycle per instruction (CPI) is 1.5.	1. Has complex instructions that take up multiple clocks for execution. The average clock cycle per instruction (CPI) is in the range of 2 and 15.
2. Performance is optimized with more focus on software.	2. Performance is optimized with more focus on hardware.
3. It has no memory unit and uses separate hardware to implement instructions.	3. It has a memory unit to implement complex instructions.
4. It has a hard-wired unit of programming.	4. It has a microprogramming unit.
5. The instruction set is reduced i.e. it has only a few instruction in the instruction set. Many of these instructions are very primitive.	5. The instruction set has a variety of different instructions that can be used for complex operations
6. The instruction set has a variety of different instructions that can be used for complex operations.	6. Has many different addressing modes and can thus be used to represent higher-level programming language statements more efficiently.
7. Complex addressing modes are synthesized using the software.	7. Already supports complex addressing modes.
8. Multiple register sets are present.	8. Only has a single register set.
9. Processors are highly pipelined.	9. They are normally not pipelined or less pipelined.
10. The complexity of RISC lies with the compiler that executes the program.	10. The complexity lies in the microprogram.
11. Execution time is very less.	11. Execution time is very high.
12. RISC architecture is used in high-end applications such as video processing, telecommunications and image processing.	12. CISC architecture is used in low-end applications such as security systems, home automation, etc.

Direct Video RAM Accessing

- Not important

Memory module

- Not important

Introduction to Assembly Language

Program structure and its components

- Read from Assembly Language Programming and Organization of the IBM PC
- 4.1: Assembly Language Syntax
  - 4.1.1: Name Field
  - 4.1.2: Operation Field
  - 4.1.3: Operand Field
  - 4.1.4: Comment Field
- 4.2: Program Data
- 4.3: Variables
  - 4.3.1: Byte Variables

    - 4.3.2: Word Variables
    - 4.3.3: Arrays
- 4.7: Program Structure
    - 4.7.1: Memory Models
    - 4.7.2: Data Segment
    - 4.7.3: Stack Segment
    - 4.7.4: Code Segment
    - 4.7.5: Putting It Together
- 4.10: Creating and Running a Program (*would like to think of it as OPTIONAL*)
    - The Source Listing File
    - The Cross-Reference File

Few basic instructions

- 4.5: A Few Basic Instructions
    - 4.5.1: MOV and XCHG
    - 4.5.2: ADD, SUB, INC and DEC
    - 4.5.3: NEG
- At the start of Main Procedure:

  MOV AX, @DATA
MOV DS, AX

  Why do we need this?
    - *Read Program Segment Prefix* (article 4.11)
- Before closing the Main Procedure:

  MOV AH, 4CH
INT 21H

  This returns control to DOS

Input/output instruction

- 4.8: Input and Output Instructions
    - 4.8.1: INT 21h
- 4.11: Displaying a String
    - The LEA Instruction
    - Program Segment Prefix

Flag Register and Flow Control

Read from Assembly Language Programming and Organization of the IBM PC

- 5.1: The FLAGS Register
- 5.2: Overflow
- 5.3: How Instructions Affect The Flags
- 6.1: An Example of a Jump
- 6.2: Conditional Jumps
    - Range of a Conditional Jump
    - How the CPU Implements a Conditional Jump
    - The CMP Instruction
    - Interpreting the Conditional Jumps
    - Signed Versus Unsigned Jumps
    - Working with Characters

- 6.3: The JMP Instruction
- 6.4: High-Level Language Structures
  - 6.4.1: Branching Structures (*not that important, I think*)
  - 6.4.2: Looping Structure

Logic and Arithmetic Operation

Logic

- Read from Assembly Language Programming and Organization of the IBM PC
- 7.1: Logic Instructions
  - 7.1.1: AND, OR and XOR Instructions
  - 7.1.2: NOT Instruction
  - 7.1.3: TEST Instruction

Shift and rotate instruction

- 7.2: Shift Instructions
  - 7.2.1: Left Shift Instructions
  - 7.2.2: Right Shift Instructions
- 7.3: Rotate Instructions

Multiplication and division instructions

- 9.1: MUL and IMUL
- 9.3: DIV and IDIV
- 9.4: Sign Extension of the Dividend

Arrays and Data Structure

Arrays and related addressing models

- Read from Assembly Language Programming and Organization of the IBM PC
- 10.1: One-Dimensional Arrays
  - The DUP Operator
  - Location of Array Elements
- 10.2: Addressing Modes
  - 10.2.1: Register Indirect Mode
  - 10.2.2: Based and Indexed Addressing Modes
  - 10.2.3: The PTR Operator and the LABEL Pseudo-op
- 10.4: Two-Dimensional Arrays
  - How Two Dimensional Arrays Are Stored
  - Locating an Element in a Two Dimensional Array
- 10.5: Based Indexed Addressing Mode
- 10.7: The XLAT Instruction

Stack

- 8.1: The Stack

Procedures

- 8.3: Terminology of Procedures

- Procedure Declaration
- RET
- Communication Between Procedures
- Procedure Documentation (*optional*)

- 8.4: CALL and RET

String Manipulation

Read from Assembly Language Programming and Organization of the IBM PC

- 11.1: The Direction Flag
- 11.1: The Direction Flag
- 11.2: Moving a String
- 11.3: Store String
- 11.4: Load String
- 11.5: Scan String
- 11.6: Compare String
  - Finding a Substring of a String
- 11.7: General Form of the String Instructions

Something Extra:

- What happens after you turn on the power of your computer?
  - https://www.geeksforgeeks.org/what-happens-when-we-turn-on-computer/
- What are the requirements of connecting multiple PC's to work together?
- How to add 256 bit numbers in a 64 bit microprocessor?

Resources:

1. https://www.elprocus.com/evolution-of-microprocessor-with-applications/
2. https://www.techwalla.com/articles/basic-components-of-microprocessors
3. https://www.tutorialspoint.com/microprocessor/microprocessor_8257_dma_controller.htm
4. http://inputoutput5822.weebly.com/direct-memory-access.html
5. https://en.wikipedia.org/wiki/Direct_memory_access#Modes_of_operation
6. https://whatis.techtarget.com/definition/coprocessor
7. https://en.wikipedia.org/wiki/Coprocessor
8. https://www.tutorialspoint.com/microprocessor/microprocessor_8087_numeric_data_processor.htm
9. https://en.wikipedia.org/wiki/Reduced_instruction_set_computer
10. https://www.studytonight.com/computer-architecture/risc-cisc-processors
11. https://www.watelectronics.com/what-is-risc-and-cisc-architecture/
12. https://www.geeksforgeeks.org/computer-organization-risc-and-cisc/
13. https://www.elprocus.com/difference-between-risc-and-cisc-architecture/
14. https://www.elprocus.com/what-is-risc-and-cisc-architecture-and-their-workings/
15. https://en.wikipedia.org/wiki/PowerPC
16. https://www.techopedia.com/definition/28749/powerpc
17. https://en.wikipedia.org/wiki/Complex_instruction_set_computer
18. https://en.wikipedia.org/wiki/Explicitly_parallel_instruction_computing