Computer Organization

(Ders Notları)

Dr. Cahit Karakuş

2017

İçindekiler

1. Tanımlar

Hertz: Frekans birimidir. Bir saniyedeki peryodik döngü sayısı. Bir işlemci döngüsü, işlemcinin saat darbeleri ile belirlenir. T=1/f, Birim: Hz, KHz, MHz, GHz, THz, ...

MIPS (Millions of Instructions per Second): Bir işlemcinin bir saniyede kaç milyon komut çalştırdığını ölçer. Tek döngülü işlemci, komut setindeki her komutu tek bir saat peryodunda tamamlama kapasitesine sahiptir. Single-cycle işlemcinin problemleri multi-cycle ve pipelined işlemciler ile ortadan kaldırılmıştır. Single-cycle işlemcilerin bir dezavantajı da her komutu eşit zamanda tamamlamasıdır. Ayrıca mantık kapı geçikmelerinden dolayı clock frekansı artırılamaz. Paralel komut yürütme işlemi, her işlemci döngüsü başına işlediği komut sayısını artırır.

Paralel komut yürütümü çok daha fazla transistör gerektirir; bu nedenle transistör sayısındaki artış, saat çevrim hızı arttıkça saniye başına daha fazla yürütülebilen komut sayısını arttırmıştır. Daha büyük bir transistor sayısı, komut başına işlenen bilginin miktarını arttırarak mikro işlemcinin grafik gibi özel bilgileri işleme hızını arttıran özel komutların eklenmesini sağlar. Görev paylaşım esasına göre çalışan çoklu işlemciler, her saat çevrimi için birden fazla komut işler. **GIPS** (Giga Instructions per Second) bir saniyede işlenen bir milyar komut sayısı. 1 bin MIPS = 1 GIP

Design Rule: because the wires and components, including transistors, on chips are drawn photographically, the pixel size of the imaging process determines the width of the wires and the size of the transistors. The size of the transistors determines how many will fit on a chip of a given size. (The optimal size of a chip depends on the chip manufacturing processes. In general, chip size increases slowly over time.) The smaller the transistors, the more will fit on the chip, determining the chip's transistor budget. The size of the transistors also determines the transistor's switching speed. Smaller transistors switch faster. One micron is one one-millionth of a meter or about 40 millionths of one inch. Finally, the power required to switch smaller transistors is less, so smaller pixels in the design rules allow the batteries in laptop computers to last longer.

Number of Transistors: The number of transistors increases as the square of decrease in design rule size. Each reduction in design rule size is chosen to about double the number of available transistors (the transistor budget). [For example: (.25micron / .18 micron) x (.25 / .18) = 2.] The gradual increase in die size (the size of the chip) also increases the number of transistors. The Itanium chip has 30 million processor transistors and 300 million memory cache transistors

Motherboard: Bilgisayarlarda mothernoard denilen ana devre kartı üzerine bir dizi chip monte edilir. CPU ve bellekler motherboard üzerine yerleştirilir.

1 Byte: 8 bitlik bellek gözünü işaret eder, gösterir. En küçük depolama birimidir.

Statik IP adresi: Servis sağlayıcı tarafından verilen ve hiç değişmeyen bir adrestir.

Dinamik IP adresi: İnternet Servis Sağlayıcısı tarafından kullanıcıya her internete bağlandığında geçici olarak atanan değişken bir ip adresidir.

Dinamik: Harekete sebep olan ve hareketi değiştiren unsurları inceler.

Statik: (sıfat) Belirli bir süre hep aynı kalan, devinimi olmayan, durağan, duruk. Değişme, gelişme, ilerleme göstermeyen, değişmeyen.

Strateji: Önceden belirlenen bir amaca ulaşmak için tutulan yolların ve uygulanan yöntemlerin tümü. Bir savaşta amaca ulaşmak için askeri kuvvetleri uygun bir biçimde kullanma sanat ve bilimi. Barışta ve savaşta benimsenen politikalara en çok desteği vermek ereğiyle siyasal, ekonomik, psikolojik ve askeri güçleri bir arada kullanma, düzenleme bilim ve sanatı.

Windows: İşletim sistemidir. Windows, ilk defa kişisel bilgisayar kullanıcılarının karşısına 20 Kasım 1985 tarihinde çıkmış olan, kullanıcılara tamamen ara yüzle yaklaşmayı prensip edinmiş bir işletim sistemi ailesidir. Bir Microsoft ürünüdür.

Operating system is a collection software routines. Operating system is a collection of software routines.

ASCII kodu, 8 bitliktir.

Intel: Pentium serisi mikroişlemcileri üreten firmadır.

Kodlayıcı (encoder), sayısal bir bilginin, başka bir sayısal bilgiye dönüştürülmesi için kullanılan lojik bir devredir. Örneğin, desimal (onluk) sayı sisteminde girilen sayısal bilgileri, binary (ikili) sayı sistemlerine dönüştürür. Alfanümerik tuş takımlarında ve klavyede kodlayıcı devreler kullanılır.

Decimal – BCD kodlayıcı

10'luk sayı sisteminde ki kodları, bcd (ikili sayı sistemi) koduna dönüştürür.

Kod Çözücü (decoder) Devreler

Kod çözücü(decoder), kodlanmış bilgileri anlaşılır hale dönüştürmek için kullanılır. Örneğin cep telefonumuza gelen mesajları 2'lik sayı sisteminde anlayamayız. Gelen bilgiler çözümlenerek metin formatı haline getirilir. Bilgisayarda anakart, diğer mikroişlemcilerde adresleme amacıyla kullanılır.

1.1. CPU

CPU kontrol ünitesi ve aritmetik mantık ünitesinden oluşur.

Bir CPU'nun dış dünya ile iletişimi 3 şekilde olur:

- Data Bus
- Address Bus
- Control Bus

ALU: Aritmetik Logic Unit

Multitasking: İki veya daha fazla görev, iş veya programın aynı anda çalışmasına izin veren işlem ve belle yönetimi hizmetleri sağlar.

Multiprocessising: CPU nun hızını artırmak amacıyla işleri, görevleri ya da programları bölerek farklı işlemcilerde çalışmasına izin veren CPU hizmetleri sağlar.

Flags: CPU'nun işlevselliğini belirleyen Status Register ismiyle anılır.

Processing system:

Bus: CPU'nun bellek ve çevre birimleri arasındaki iletişim hatlarına verilen addır.

1.2. Computer

A computer system is best characterized in terms of structure— the way in which components are interconnected, and function— the operation of the individual components. Furthermore, a computer's organization is hierarchical. Each major component can be further described by decomposing it into its major subcomponents and describing their structure and function.

For clarity and ease of understanding, this hierarchical organization is described in this lecture from the top down:

- Computer system: Major components are processor, memory, I/O.
- Processor: Major components are control unit, registers, ALU, and instruction execution unit.
- Control unit: Provides control signals for the operation and coordination of all processor components. Traditionally, a microprogramming implementation has been used, in which major components are control memory, microinstruction sequencing logic, and registers. More recently, microprogramming has been less prominent but remains an important implementation technique. A computer accepts and processes data using a set of stored instructions.

To illustrate the concepts and to tie them to real- world design choices that must be made, two processor families have been chosen as running examples:

- **Intel x86 architecture:** The x86 architecture is the most widely used for nonembedded computer systems. The x86 is essentially a **complex instruction set computer (CISC**) with some RISC features. Recent members of the x86 family make use of superscalar and multicore design principles. The evolution of features in the x86 architecture provides a unique casestudy of the evolution of most of the design principles in computer architecture.
- **ARM:** The ARM architecture is arguably the most widely used embedded processor, used in cell phones, iPods, remote sensor equipment, and many other devices. The ARM is essentially a **reduced instruction set computer (RISC).** Recent members of the ARM family make use of superscalar and multicore design principles.

Function:

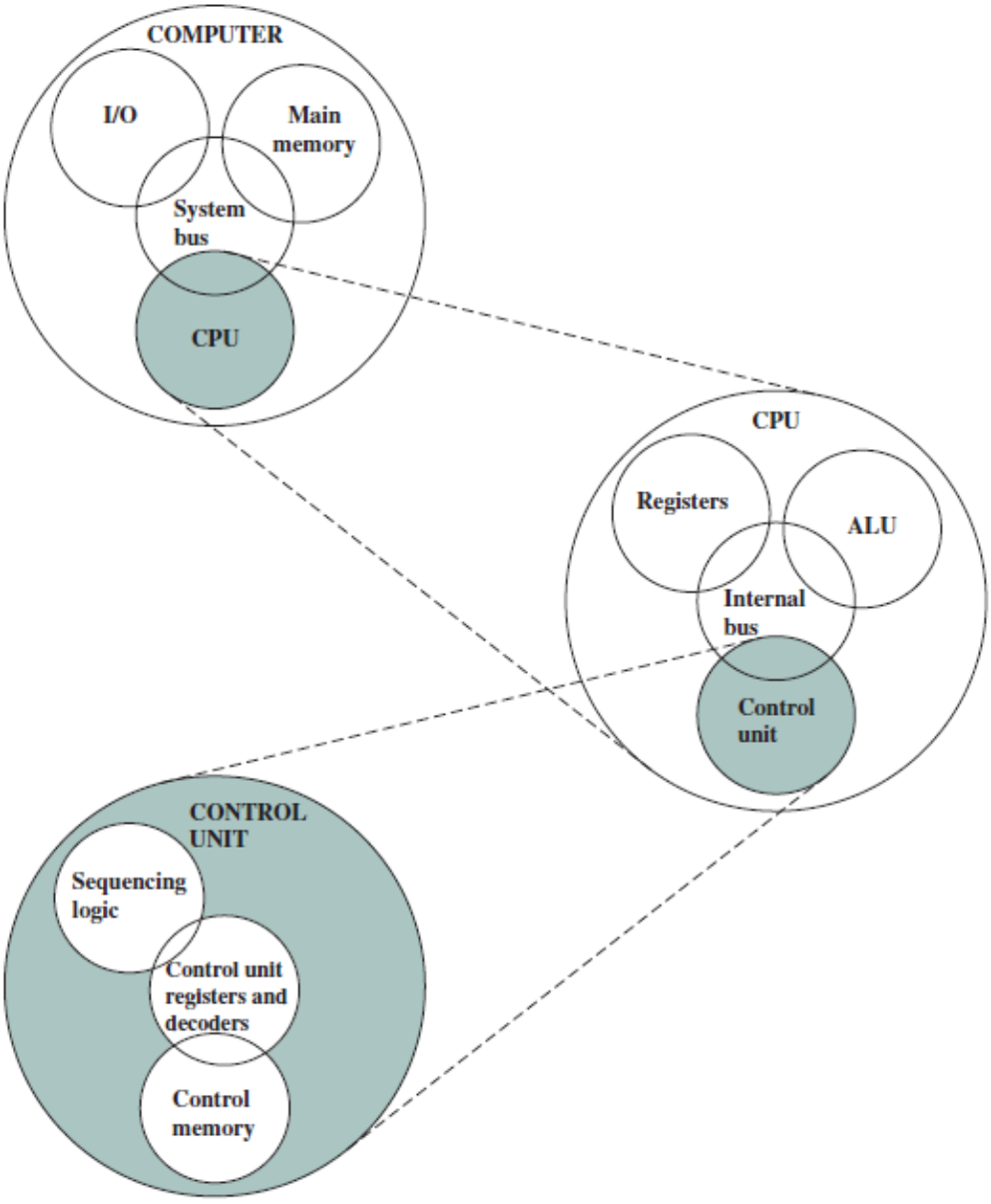
Both the structure and functioning of a computer are, in essence, simple. In general terms, there are only four basic functions that a computer can perform:

- **Data processing:** Data may take a wide variety of forms, and the range of processing requirements is broad. However, we shall see that there are only a few fundamental methods or types of data processing.
- **Data storage:** Even if the computer is processing data on the fly (i.e., data come in and get processed, and the results go out immediately), the computer must temporarily store at least those pieces of data that are being worked on at any given moment. Thus, there is at least a short- term data storage function. Equally important, the computer performs a long- term data storage function. Files of data are stored on the computer for subsequent retrieval and update.
- **Data movement:** The computer's operating environment consists of devices that serve as either sources or destinations of data. When data are received from or delivered to a device that is directly connected to the computer, the process is known as *input– output* (*I*/*O*), and the device is referred to as a *peripheral*. When data are moved over longer distances, to or from a remote device, the process is known as *data communications*.
- **Control:** Within the computer, a control unit manages the computer's resources and orchestrates the performance of its functional parts in response to instructions. The preceding discussion may seem absurdly generalized. It is certainly possible, even at a top level of computer structure, to differentiate a variety of functions, but to quote. There is remarkably little shaping of computer structure to fit the function to be performed. At the root of this lies the general- purpose nature of computers, in which all the functional specialization occurs at the time of programming and not at the time of design.

Structure:

We now look in a general way at the internal structure of a computer. We begin with a traditional computer with a single processor that employs a microprogrammed control unit, then examine a typical multicore structure. S***imple single-processor computer*** Figure 1.1 provides a hierarchical view of the internal structure of a traditional single-processor computer. There are four main structural components:

- **Central processing unit (CPU):** Controls the operation of the computer and performs its data processing functions; often simply referred to as **processor**.
- **Main memory:** Stores data.



The Computer: Top-Level Structure

- **I/O:** Moves data between the computer and its external environment.
- **System interconnection:** Some mechanism that provides for communication among CPU, main memory, and I/O. A common example of system interconnection is by means of a **system bus**, consistingof a number of conducting wires to which all the other components attach.

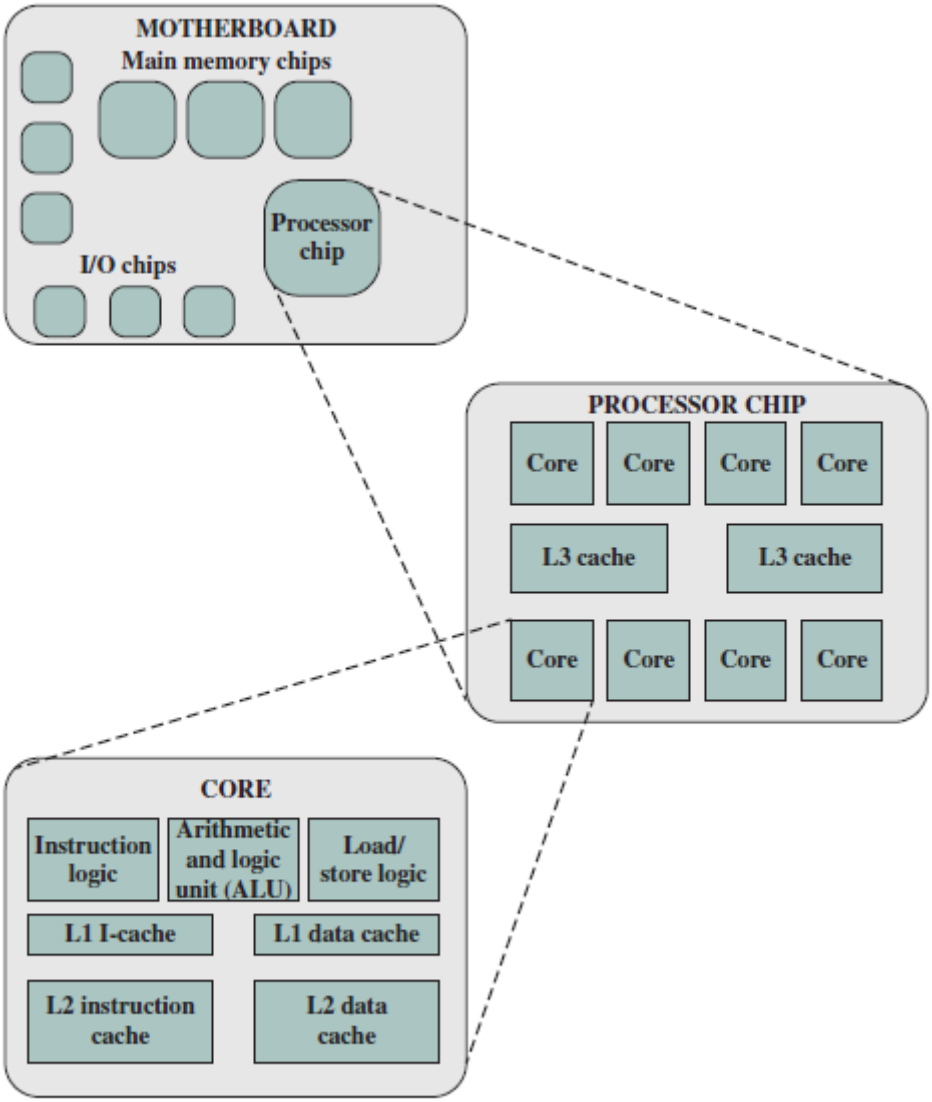
Each of these components will be examined in some detail in Part Two. However, for our purposes, the most interesting and in some ways the most complex component is the CPU. Its major structural components are as follows:

- **Control unit:** Controls the operation of the CPU and hence the computer.
- **Arithmetic and logic unit (ALU):** Performs the computer's data processing functions.
- **Registers:** Provides storage internal to the CPU.
- **CPU interconnection:** Some mechanism that provides for communication among the control unit, ALU, and registers.

Multicore computer structure:

Multicore computer structure As was mentioned, contemporary computers generally have multiple processors. When these processors all reside on a single chip, the term *multicore computer* is used, and each processing unit (consisting of a control unit, ALU, registers, and perhaps cache) is called a *core*. To clarify the terminology, this text will use the following definitions.

- **Central processing unit (CPU):** That portion of a computer that fetches and executes instructions. It consists of an ALU, a control unit, and registers. In a system with a single processing unit, it is often simply referred to as a *processor*.
- **Core:** An individual processing unit on a processor chip. A core may be equivalent in functionality to a CPU on a single- CPU system. Other specialized processing units, such as one optimized for vector and matrix operations, are also referred to as cores.
- **Processor:** A physical piece of silicon containing one or more cores. The processor is the computer component that interprets and executes instructions. If a processor contains multiple cores, it is referred to as a **multicore processor**.



Simplified View of Major Elements of a Multicore Computer

The motherboard contains a slot or socket for the processor chip, which typically contains multiple individual cores, in what is known as a *multicore processor*. There are also slots for memory chips, I/O controller chips, and other key computer components. For desktop computers, expansion slots enable the inclusion of more components on expansion boards. Thus, a modern motherboard connects only a few individual chip components, with each chip containing from a few thousand up to hundreds of millions of transistors.

A processor chip that contains eight cores and an L3 cache. Not shown is the logic required to control operations between the cores and the cache and between the cores and the external circuitry on the motherboard. The figure indicates that the L3 cache occupies two distinct portions of the chip surface. However, typically, all cores have access to the entire L3 cache via the aforementioned control circuits. The processor chip does not represent

any specific product, but provides a general idea of how such chips are laid out.Next, we zoom in on the structure of a single core, which occupies a portion of the processor chip. In general terms, the functional elements of a core are:

- Instruction logic: This includes the tasks involved in fetching instructions, and decoding each instruction to determine the instruction operation and the memory locations of any operands.
- **Arithmetic and logic unit (ALU):** Performs the operation specified by an instruction.
- **Load/store logic:** Manages the transfer of data to and from main memory via cache.

The core also contains an L1 cache, split between an instruction cache (I- cache) that is used for the transfer of instructions to and from main memory, and an L1 data cache, for the transfer of operands and results. Typically, today's processor chips also include an L2 cache as part of the core. In many cases, this cache is also split between instruction and data caches, although a combined, single L2 cache is also used.

Keep in mind that this representation of the layout of the core is only intended to give a general idea of internal core structure. In a given product, the functional elements may not be laid out as the three distinct elements shown in Figure 1.2, especially if some or all of these functions are implemented as part of a microprogrammed control unit.

Bilgisayar, giriş birimleri ile dış dünyadan aldıkları veriler üzerinde aritmetiksel ve mantıksal işlemler yaparak işleyen ve bu işlenmiş bilgileri çıkış birimleri ile bize ileten, saklayan; donanım (hardware) ve yazılım (software) dan oluşan elektronik bir makinedir. Gerçek dünyada bilgisayarların tümü sayısaldır.

1981 yılında 1 GB'lık depolama biriminin maliyeti 300 bin doları geçerken, bu rakam yıldan yıla azaldı. 1987 yılında 500 bin dolara, 2000 yılında 10 dolara ve 2012'de 1 doların dahi altına düştü.

Computers are classified based on their technology, function, physical size, performance and cost. The categories of computers include:

- Main-frame bilgisayarlar : Aynı anda yüzlerce tkullanıcı aracılığıyla kullanılan büyük bilgisayar sistemleridir. İşlem gücü oldukça yüksektir. Genellikle büyük şirketlerde, bilgi işlem merkezlerinde, araştırma kurumlarında ve üniversitelerde kullanılırlar.
- Midi (orta) bilgisayarlar : Main-frame bilgisayarlarla aynı işlevi gören daha küçük bilgisayar sistemleridir. Orta boy işletmeler tarafından tercih edilirler.
- PC (Personal Computer) : Kişisel bilgisayarlar (masa üstü, diz üstü), el terminalleri ve akıllı telefonlar olarak adlandırılan teknolojilerin kullanıldığı bilgisayarlardır. Network (ağ) iletişim ortamlarında client (müşteri) ve internet servis sağlayıcısı üzerinden

birbirlerine bağlanabilmektedirler. Ofis otomasyonunda, eğitimde, yayın işlerinde, küçük işletmelerin ticari hesaplarının ve personel kayıtlarının tutulmasında etkin biçimde kullanılırlar.

Personal computers, Handheld computers, Mainframes, Supercomputers, Akıllı teknolojilier
Hardware: Bilgisayarın donanım kısımlarına verilen addır.
Software: Bilgisayarın yazılım ısmına verilen addır.

Bir bilgisayarın temel fonkisyonları:

- Store data
- Accept data
- Process data

Assembler yazılımdır, donanım değildir.
Assembly Language programlama dili bir bilgisayarın makine diline en yakın talimatı verir.

1.3. Bellek

RAM: Random Access Memory. Read /write memory. Primary memory.
ROM: Read Only Memory. ROM'daki veriler uçucu değildir, elektrik gücü olmadan veriler saklanır.
Register: Verileri geçici olarak kayıt eder, yazılımın belirtildiği şekilde iletilir.

BIOS:
Bilgisayarın içindeki tüm veri akışını düzenleyen, ve bileşenler arasındaki koordinasyonu sağlayan anakartların en önemli temel parçalarından birisi BIOS'tur. (BASIC INPUT OUTPUT SYSTEM - TEMEL GİRİŞ ÇIKIŞ SİSTEMİ). BIOS sadece anakartlarda değil ekran kartı gibi bileşenler de kullanılır. Üzerindeki donanımın teknik özelliklerine ait bilgileri barındırır.

BIOS, genel olarak bilgisayarın açılabilmesi için gerekli bir yazılımdır. Bu yazılım BIOS Chipi içinde tutulur. Sonuçta bir yazılım olduğu için istenirse silinebilir ve tekrar yüklenebilir. BIOS ilgili donanımın beynidir. BIOS'u zarar gören anakart işe yaramaz. BIOS Chip'inin değiştirilmesi veya tekrar yazılımının yüklenmesi gerekir. BIOS, bilgisayarın sabit disk ya da disket sürücü gibi kaynaklardan açılabilmesini sağlaması dışında POST (Power On Self Test), açılışta bilgisayarın kendini test etmesi işleminden de sorumludur. BIOS görevini yerine getirebilmek için, bazı donanım bilgilerin ihtiyaç duyar. Bu bilgiler, bilgisayar kapalı da olsa silinmezler. BIOS verilerini tutan yarı iletkene CMOS denir. 64 byte' lık kapasitesi içinde BIOS için gerekli bilgileri saklayabilmek için minik bir pil kullanır.

1.4. İkili sayı sistemi

- **Ondalık** (taban 10).

HEX - hexadecimal (taban 16). Hex: 4 bit grup: 0….9,A,B,C,D,E,F. Headecimal sayı sistemi 16 lık temelde çalışır.
(1066)hex=(0001 0000 0110 0110)b

BIN: ikili -binary (taban 2). Dijital sistemler genellikle ikili sayı sisteminde çalışır.

- **OCT**: octal (base 8).
- **SIGNED:** signed decimal (base 10).
- **UNSIGNED:** unsigned decimal (base 10).
- **CHAR:** ASCII char code; 256 sembol (0…255 ondalık sayı ya da 00h….FFh hex olarak tanımlanır).

- Byte: 8 bit veriyi işaret eder
- Bit/sec: Veri transfer birimi: 1 saniyede transfer edilen bit sayısı: veri miktarı

Kilo Byte	Kb	2^{10} Byte
Mega Byte	Mb	2^{20} Byte
Giga Byte	Gb	2^{30} Byte
Tera Byte	Tb	2^{40} Byte
Peta Byte	Pb	2^{50} Byte
Exa Byte	Eb	2^{60} Byte
Zetta Byte	Zb	2^{70} Byte
Yotta Byte	Yb	2^{80} Byte

0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

How to convert numbers between decimal and binary.

A binary number looks like: **01001010.** Each digit of a binary number is based on 2 to the power of x .

2 to the power of 0 = **1**

2 to the power of 1 = **2**

2 to the power of 2 = **4**

2 to the power of 3 = **8**

2 to the power of 4 = **16**

2 to the power of 5 = **32**

2 to the power of 6 = **64**

2 to the power of 7 = **128**

2 to the power of 8 = **256**

What is the decimal equivalent of 01001010?

- All digits that are 0 remain 0, and are only useful as position placeholders.
- All digits that are assigned a value of 1 have a decimal value that is equal to the power (2^x) of their position within the chart.

It's helpful to **create a chart** when converting between decimal and binary

Decimal 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1

Binary 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0

Demonstrate how to convert numbers between decimal and binary. Let's convert the decimal number 74 to binary. First, find the biggest power of 2 that is less than 74.

- 2 to the power of 6 = 64
- Next, subtract 74-64 = 10
- Repeat the process to find the biggest power of 2 that is less than 10. Right, that would be 2 to the power of 3 = 8
- Next, subtract 10-8 = 2
- Repeat the process to find the biggest power of 2 that is less than 2. Right, that would be 2 to the power of 1 = 2

128 | 64 | 32 | 16 | 8 | 4 | 2 | 1

---

0 | **1** | 0 | 0 | **1** | 0 | **1** | 0

ON ON ON

- So decimal $(74)_d$ = $(01001010)_b$

Powers of Two

n	***2^n***	***n***	***2^n***	***n***	***2^n***
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024 (1K)	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096 (4K)	20	1,048,576 (1M)
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

Ondalık (decimal) sayı sisteminden ikili (binary) ve onaltılık (hex) sayı sistemine dönüştürme:

$a_n\ 2^n + a_{n-1}\ 2^{n-1} + \cdots + a_1\ 2^1 + a_0\ 2^0$

İkili sayıların ondalık sayılara dönüştürülmesi:

$(100011)_2 = 2^5 + 2^1 + 2^0 = 32+2+1 = (35)_{10} = (23)_{16}$

$(1010)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= 8 + 0 + 2 + 0 = (10)_{10}$

Ondalıklı binary sayıların decimal sayılara dönüştürülmesi:

$(111{,}101)_2 = 2^2+2^1+2^0+2^{-1}+2^{-3} = 4+2+1+1/2+1/8 = 7{,}625$

Decimal sayıların binary sayılara çevrilmesi: $(172)_{10} = (128+32+8+4)_{10} = (2^7+2^5+2^3+2^2)_{10} = (1010\ 1100)_2 = (AC)_{16}$

Ondalıklı decimal sayıların binary sayılara dönüştürülmesi:

$(10, 75)_{10} = ?$

Tam kısmı: $(2^3+2^1)_{10} = (1010)_2$, Ondalıklı kısım: $2^{-1} = 1/2 = 0{,}5$, $2^{-2} = 1/4 = 0{,}25$,

$(10{,}75)_{10} = (1010{,}11)_2$

$0.123 = 1 \times 10^{-1} + 2 \times 10^{-2} + 3 \times 10^{-3}$

$(0.101)_2 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 0.625$

$(110.011)_2 = 4 + 2 + 0.25 + 0.125 = 6.375$

0.5625 × 2 = 1.125	first bit = 1
0.125 × 2 = 0.25	second bit = 0
0.25 × 2 = 0.5	third bit = 0
0.5 × 2 = 1.0	fourth bit = 1

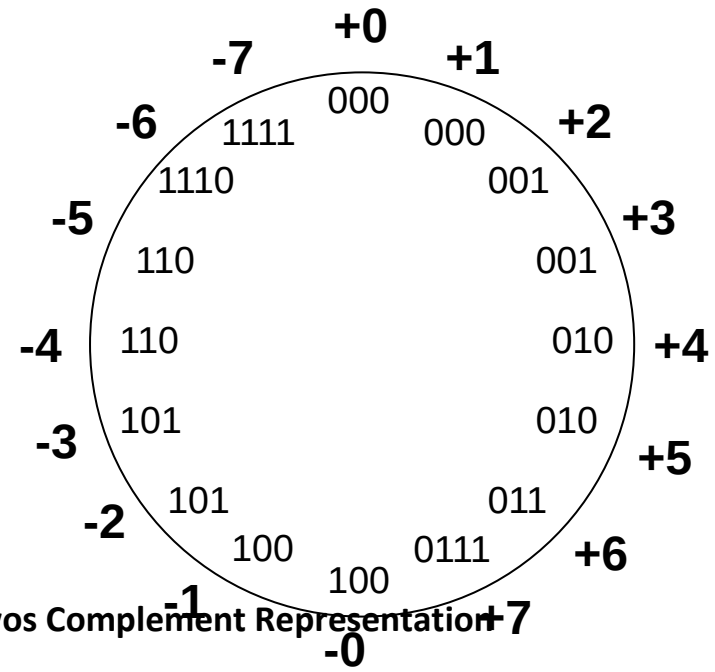
Negative number:

If the number is negative, complement all bits and add by 1

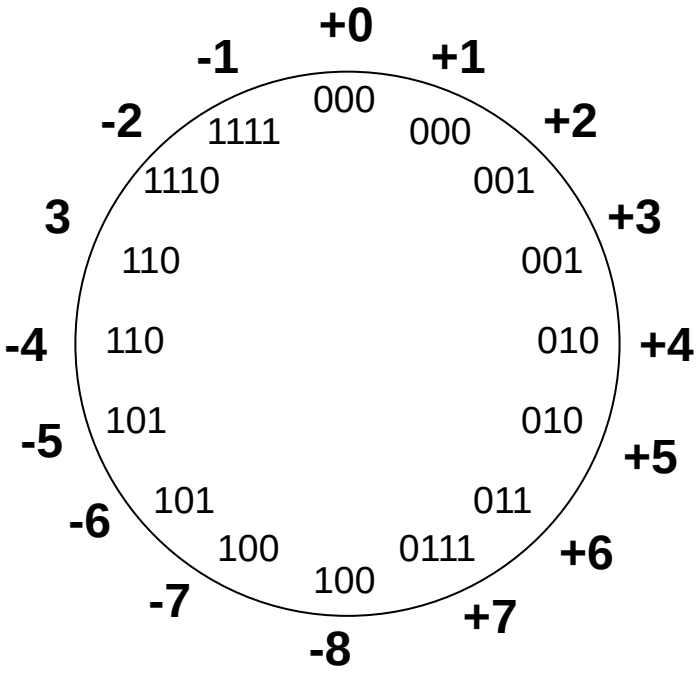
-6 => $\overline{0000\ 0110}$ + 1 = 1111 1001 + 1 = FAh

Decimal	Binary	Hex
-128	1000 0000 b	80h
-127	1000 0001b	81h
-126	1000 0010b	82h
…		…
-2	1111 1110b	FEh
-1	1111 1111b	FFh
0	0000 0000b	00h
1	0000 0001b	01h
…		…
+127	0111 1111b	7Fh

Sign-Magnitude Representation



Twos Complement Representation



Binary Arithmetic

0 + 0 = 0, 0 and carry 0
1 + 0 = 1, 1 and carry 0
0 + 1 = 1, 1 and carry 0
1 + 1 = 10, 0 and carry 1
1 + 1 + 1 = 11, 1 and carry 1

11 + 1 = 100

101 + 1 = 110

```
                 11 1            11<--- Carry bits
   1001101   1001001   1000111
+  0010010 + 0011001 + 0010110
    ---------       ---------      ---------
   1011111   1100010   1011101
```

Bit, Bit/San

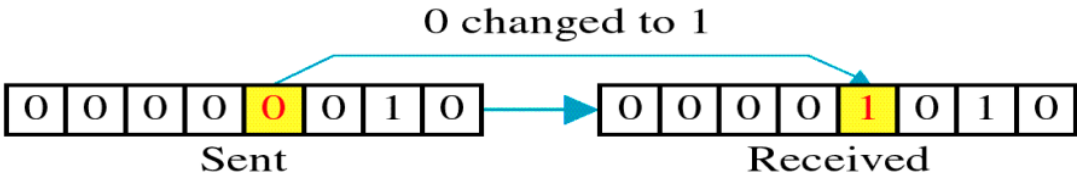
- **Bit:** Dijital elektronikte ve binary sayı sisteminde sadece 0 ve 1 değerleri vardır. Tüm işlemler bu iki değer üzerinden yapılır. 0 ya da 1 bilgisinin her birine bit denir. Bit→0/1 den oluşan bilgi
- Bits are the units used to describe an amount of data in a network
  - 1 kilobit (Kbit) = 1 x 10^3 bits = 1,000 bits
  - 1 megabit (Mbit) = 1 x 10^6 bits = 1,000,000 bits
  - 1 gigabit (Gbit) = 1 x 10^9 bits = 1,000,000,000 bits
- ***Bit/Saniye:*** Bit/sec→1 sn. ye de bir noktadan diğer noktaya iletilen bilgi. **BPS (Bit Per Second); S**aniyede iletilen bit sayısına BPS denir. Veri işleme hızı ve veriiletim hızı birimidir.
- Seconds are the units used to measure time
  - 1 millisecond (msec) = 1 x 10^{-3} seconds = 0.001 seconds
  - 1 microsecond (msec) = 1 x 10^{-6} seconds = 0.000001 seconds
  - 1 nanosecond (nsec) = 1 x 10^{-9} seconds = 0.000000001 seconds
  - Bits per second are the units used to measure channel capacity/bandwidth and throughput
  - bit per second (bps)
  - kilobits per second (Kbps)
  - megabits per second (Mbps)

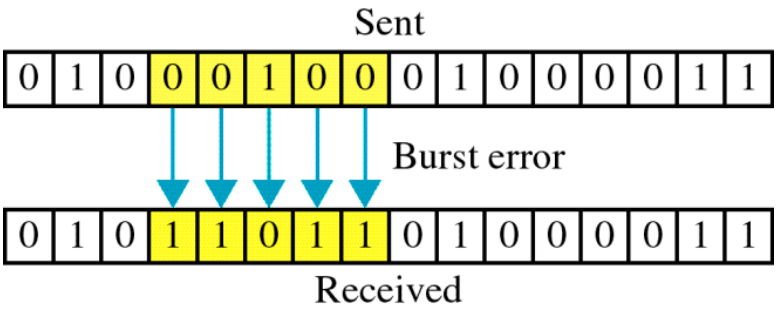
Byte, Baud Rate

- ***Byte: E***lektronik ve bilgisayar bilimlerinde genellikle 8 bitlik dizilim boyunca 1 veya 0 değerlerini bünyesine alan ve kaydedilen bilgilerin türünden **bağımsız bir bellek ölçüm birimidir.**
- **Bit** terimi belleğin 8 bitlik bir değerini işaretleyen ya da tanımlayan en küçük birimi olarak tanımlanmıştır. Daha sonra, 1956'da, 6 Bite'tan 8 Bite geliştirilmiştir. Bite, bit ile karıştırılmaması için daha sonra **Byte'**a çevrilmiştir. Diğer bir kelime açıklamasına göre de, Byte, "by eight"in (Türkçe'de *sekiz kez* veya *sekiz ile*) kısaltılmış halidir. Byte→bellekte 8bitlik adres gözü ya da bellek boyutu tanımlar tanımlanır. 1Gbyte=2^{10}Mbyte=2^{20}Kbyte=2^{30}byte
- ***Baud Rate:*** Data iletiminde modülatör çıkışında bir saniyede meydana gelen sembol (baud) değişikliğine baud hızı denir. Baud hızı baud/sn ile gösterilir. Baud hızı sinyalin anahtarlama hızını gösterir.
- Örnek: Bir veri iletim hattının iletim hızı 4800 baud/sn olsun. Bu iletim her baud 4 bitle kodlanmış bilgi içeriyorsa bps olarak hızımız 4800*4=19200 bps olur.
- Baud Rate'i kullanmadaki amaç band genişliğini daha verimli kullanmak.

BER

- **BER: Bit Error Rate (Bit Hata Oranı):** Sayısal bilgi iletiminde gönderilen veri içindeki bozulan ya da yanlış algılanan bit oranını ifade eder. BER=Gönderilen hatalı Bit Sayısı / Gönderilen Toplam Bit Sayı.
- Örnek: BER=10-6 olduğuna göre 1 milyon bit gönderildiğinde kaç bit hatalı gitmiş olur?
- BER=10-6=1/10^6= Gönderilin hatalı Bit Sayısı / Gönderilen Toplam Bit Sayı 1milyon bitte 1 bit hatalı gitmiştir.
- Örnek: 512 000 000 bit gönderildiğinde 16 bit hata meydana geliyorsa bit-error oranı nedir?
- BER=Gönderilen Hatalı Bit Sayısı / Gönderilen Toplam Bit Sayı, BER=16/512 000 000=3,125 x 10-8
- Bit Errors; Single bit, Multiple bit, Burst
- Veri iletişimde BER olduğu belirlendiğinde veri bloğu tekrar edilir, düzeltilir ya da kurtarma, iyileştirme işlemleri yapılır.





Sıkıştırma - CODEC

Codec, Compression (sıkıştırma) ve decompression (açma) kelimelerinin birleşiminden oluşur. Ayrıca codec, analog sinyali, veri ağı üzerinden taşınabilmesi için dijital hale dönüştürür.

Kanalı verimli kullanma yöntemleri: Compression, Reducing, Coding, Modulation, Canstrator, Konuşurken beklenen aralığı değerlendirme, Abone meşgülü tablosu / abone yok tablosu hazırlama ve yayınlama. Sıkıştırmadaki amaç: Daha az çoğullama devresi, daha az band genişliği, daha az iletişim ortamı.

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

2. Mikroişlemci

Computer is a machine that can receive, transmit, store, and manipulate information. It wasn't until the '80s that computers entered our homes and - thanks to the microprocessor - really made an impact on the average person's life. Nowadays, modern microprocessors can perform extremely sophisticated operations in areas such as meteorology, aviation, nuclear physics and engineering, and take up much less space as well as delivering superior performance. Over the past 40 years, microprocessors have become faster and more powerful, yet increasingly smaller and more affordable. The first microprocessor was the Intel 4004, introduced in 1971. The 4004 was primarily used to perform simple mathematical operations in a calculator called “Busicom.” Merkezi İşlemci Birimi (CPU - Central Processing Unit) olarak da bilinen mikroişlemci, tüm bilgisayarların beynidir. Birlikte çalışan çoklu mikroişlemciler ise , veri merkezlerinin, süper bilgisayarların, iletişim sistemlerinin ve bilgisayar kontrollü diğer dijital cihazların kalbidir.

2.1. İşlemciler

Intel microprocessors

x86, orijinal Intel 8086 ile ileriye dönük olarak uyumlu model numaralarından türetilmiştir. O zamandan beri x86 yönerge setine hemen hemen geriye dönük uyumluluk ile birçok ekleme ve uzantı eklenmiştir. Intel, Cyrix, AMD, VIA ve diğer pek çok işlemciden üretildi.

80386'nın 32-bit komut kümesiyle uyumluluğundan dolayı orijinal 16-bit x86'dan ayırmak için x86-32 olarak vurgulanabilir. Yeni kişisel bilgisayarlarda ve sunucularda kullanılan çoğu x86 işlemci 64-bitlik özelliklere sahip olsa da, eski bilgisayarlarda veya sistemlerde uyumluluk problemlerinden kaçınmak için x64 terimi genellikle 64-bit yazılımları belirtmek için kullanılır; x86 terimi sadece 32-bit anlamına gelir.

Günümüzde x86 mimarisi, masaüstü ve dizüstü bilgisayarların yanı sıra sunucular ve iş istasyonları arasında gittikçe artan bir çoğunlukla bulunur. MS-DOS, Windows, Linux, BSD, Solaris ve Mac OS X gibi işletim sistemleri de dahil olmak üzere büyük miktarda yazılım platformu destekliyor. Gömülü sistemlerde mimari nispeten az ve ev aletleri ve oyuncaklar gibi düşük maliyetli işlerde 16-bit x86 yongalar daha yaygın, AMD'nin Geode ve yeni Intel Atom, bu segmentte kullanılan 32 ve 64-bit tasarım örnekleridir.

Manufacturer	Part Number	Data Bus Width	Memory Size
Intel			
8086	**16**		**1M**
8088	8		1M
80286	**16**		**16M**
80386EX	16		64M
80386DX	32		4G
80386SL	16		32M
80386SLC	16		32M + 8K cache
80386SX	**16**		**16M**
80486DX/DX2	32		4G + 8K cache
80486SX	32		4G + 8K cache
80486DX4	32		4G + 16 cache
Pentium	**64**		**4G + 16K cache**
Pentium OverDrive	32		4G + 16K cache
Pentium Pro	64		64G + 16K L1 cache + 256K L2 cache
Pentium II	64		64G + 32K L1 cache + 256K L2 cache
Pentium III	64		64G + 32K L1 cache + 256K L2 cache
Pentium 4	64		4G+32K L1 cache+ 512K L2 cache (or larger) (1T for 64-bit
extensions)			
Pentium4 D (Dual Core)	64		1T + 32K L1 cache + 2 or 4 M L2 cache
Core2	64		1T + 32K L1 cache + a shared 2 or 4 M L2 cache
Itanium (Dual Core)	**128**		**1T + 2.5 M L1 and L2 cache + 24 M L3 cache**

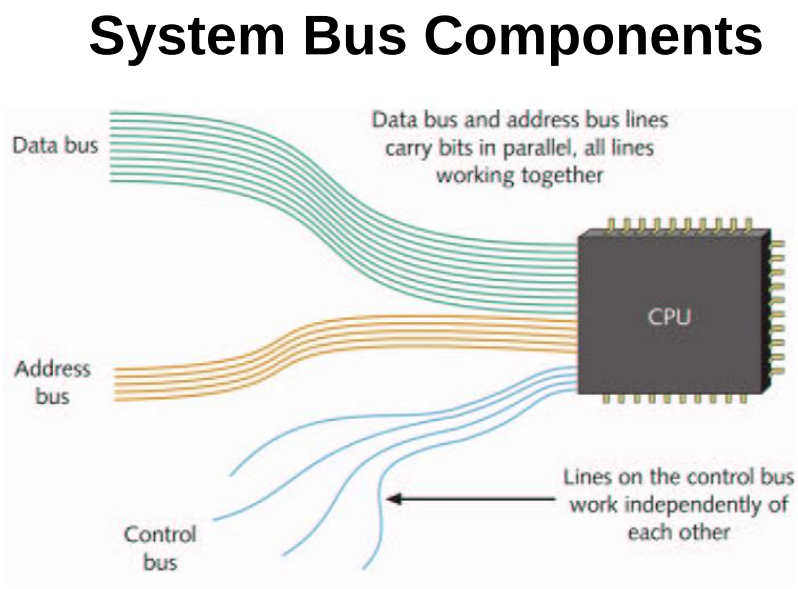
Motorola microprocessors.

Motorola 6800	**8**	**64K**
6805	8	2K
6809	8	64K
68000	16	16M
68008D	8	4M
68008Q	8	1M
68010	16	16M
68020	32	4G
68030	32	4G + 256 cache
68040	32	4G + 8K cache
68050	32	Proposed, but never released
68060	64	4G + 16K cache
Powerpc	64	4G + 32K cache

2.2. System Bus

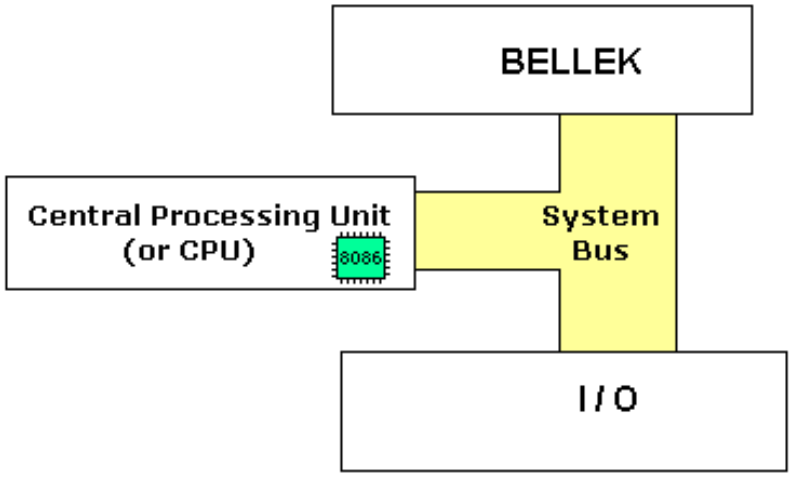
A computer consists of:

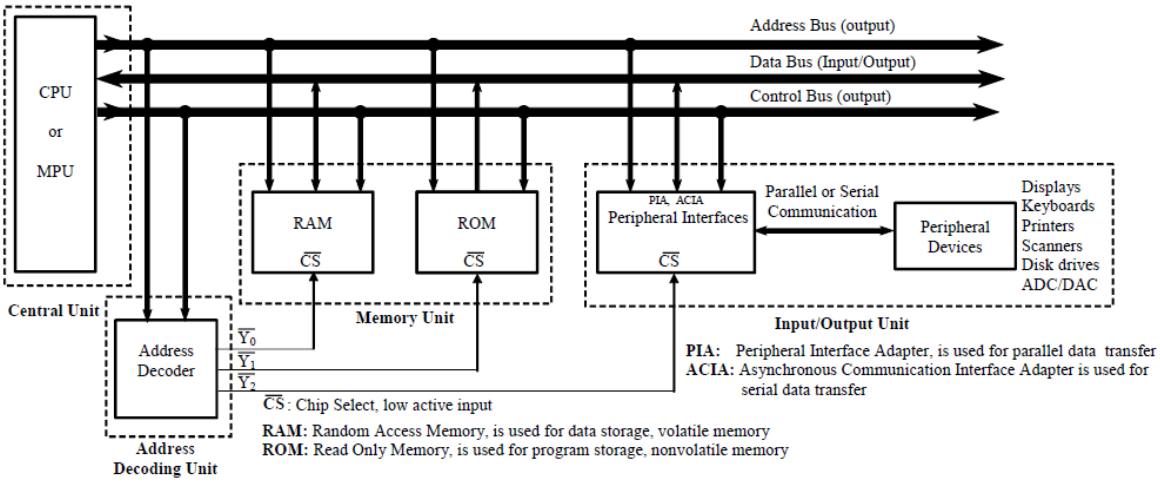
1) Central Processing Unit
2) Memory unit
3) Input/Output unit
4) Buses
5) Address decoding unit



The **SYSTEM BUS** connects the various components of a computer. A bus is a collection of wires. The busses carry address, data, and control information between the various unit. The bus consist of three parts:

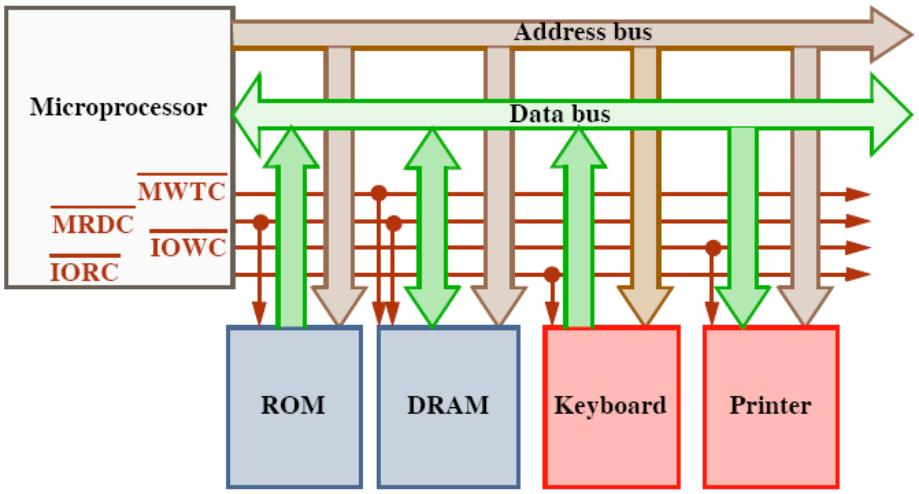
a) Address Bus (output)
b) Data Bus (input/output)
c) Control Bus (output)





The basic block diagram of a digital computer system

Bus Architecture:



- Data bus (Write /Read to memory or I/O units):
  - 8-bits (8088)
  - 16-bits (8086 /80286 /80386SX/SL/SLC/EX)
  - 32-bits (80386DX / 80486 /Pentium)
  - 64-bits (Pentium Pro /II /III)
  - 128-bits
  - 256-bits

- Control bus:

  Most systems have at least 4 control bus connections (active low)

  $\overline{\text{MRDC}}$ (Memory ReaD Control), $\overline{\text{MWRC}}$, $\overline{\text{IORC}}$ (I/O Read Control), $\overline{\text{IOWC}}$

Address Bus

- Adres yolu, bilginin bulunduğu yeri işaret eder.
- Belleğe ve I / O ekipmanını adreslemek için sisteme dahil edilmiştir.
- Çeşitli mikroişlemcilerdeki adres veri yolları yalnızca genişliklerinde (sayılar) farklılık gösterir.
- Çoğu adres veriyolu, normal mikroişlemci çalışması sırasında bir ara yüksek empedans durumuna geçecek olan üç durumlu bağlantılardır.
- n-bit adres bus 2^n byte lokasyonunu 0 dan 2^{n-1} e adresler.

  20-bits (8086/8088)
  24-bits (80286/80386X)
  32-bits (80386DX / 80486 /Pentium)
  36-bits (Pentium Pro/II/III)

The address bus width in bits is based on the microprocessor chip family.

- Each time a bit is added to the address bus width, the amount of memory (RAM: Random Access Memory) that can be addressed is doubled.
- 4 bit addresses allow the addressing of 16 bytes of memory (and extra work is necessary to address 256 bytes of memory).
- 8 bits allow the addressing of 256 bytes of memory (and extra work is necessary to address 65,536 bytes of memory).
- 16 bits can address 65,536 bytes of memory.
- 32 bits can address 4,294,967,296 bytes of memory (about 4 billion bytes).

Bus Standards:

- ISA (Industry Standard Architecture): 8 MHz
  8-bit (8086/8088)
  16-bit (80286 - Pentium)
- EISA: 8MHz, 32 bit (older 386 and 486 machines)
- PCI (Peripheral Component Interconnect): 33MHz, 32 bit or 64-bit (Pentiums)
  New: PCI Express and PCI-X 533 MTS
- VESA (Video Electronic Standards Association): Runs at processor speed.
  32 bit or 64-bit (Pentiums), Only disk and video. Competes with the PCI but is not popular.

I/O :

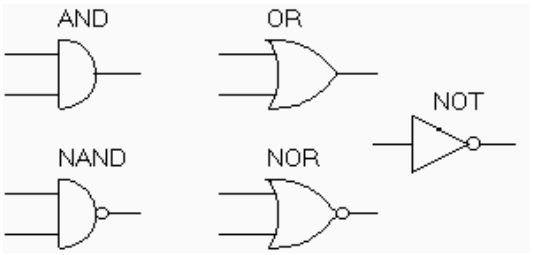
Printer, Serial Communications, Disk drivers, Mouse, Flash memory, Plooter, Keyboard, Monitor, Scanner, DVD, Backup memories.

Memory Unit

Memory is a place to where the programs and data are loaded in order to be executed.
RAM (Random Access Memory) and ROM (Read Only Memory).
RAM is read /write memory while ROMisread-onlymemory;
RAM is volatile, (the contents are lost when power is removed) while ROM is nonvolatile (the contents are not lost when power is removed).
Dynamic Ram (DRAM), Static RAM (SRAM), Cache, Read only memory (ROM), Flash Memory, ...

2.3. Memory address decoding

Lojik Kapılar



Doğruluk tablosu:

A	B	OR	AND	NOT	NOR	NAND	EXOR
		A+B	A*B	A'	(A+B)'	(A*B)'	(A')*B+A*(B')
0	0	0	0	1	1	1	0
0	1	1	0	1	0	1	1
1	0	1	0	0	0	1	1
1	1	1	1	0	0	0	0

Formüller:

A x 0=0

A x 1=A

A + 0 = A

A + 1 = 1

A x A = A

A + A=A

A x A' =0

A + A' =1

(A')$'$ =A

Sadeleştirmeler:

(A+B)= (B+A)

A x B = B x A

(A+B) +C=A+(B+C)=A+B+C

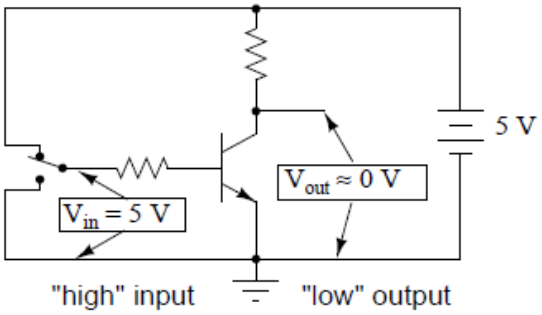
(A x B) x C=A x (B x C)=A x B x C

A + (B x C) = (A + B)(A + C)

(A + B)$'$ = A' x B'

(A x B)$'$ = A' + B'

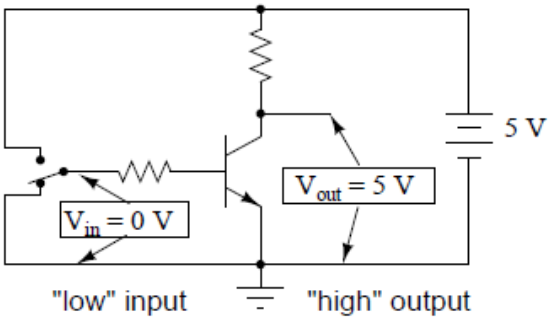
Transistor in saturation



0 V = "low" logic level (0)

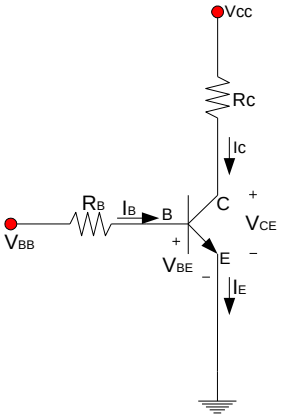
5 V = "high" logic level (1)

Transistor in cutoff



0 V = "low" logic level (0)

5 V = "high" logic level (1)



$$V_{CC} = R_C * I_C + V_{CE}$$

$$V_{BB} = R_B * I_B + V_{BE}$$

$$I_C = \beta * I_B$$

$$I_{C\,SAT} = \frac{V_{CC}}{R_C}$$

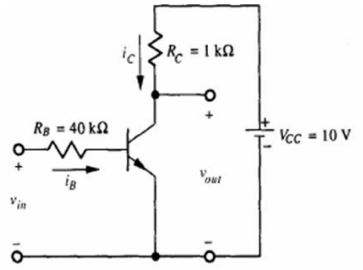
$$I_B = \frac{V_{BB} - V_{BE}}{R_B}$$

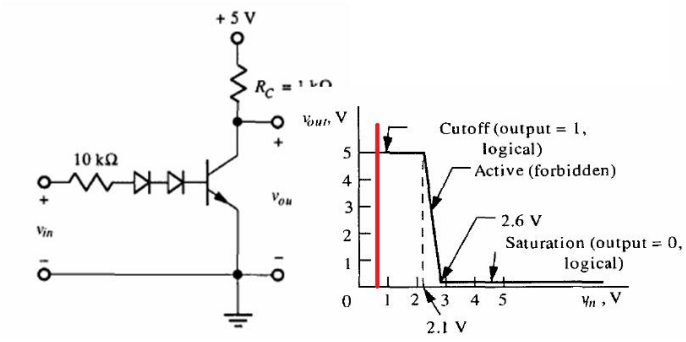
$V_{CE} \le 0\,V$ *ya da* $I_c \ge I_{c\,SAT}$; *Saturasyon* $V_{CE} = 0V$ *Olur.*

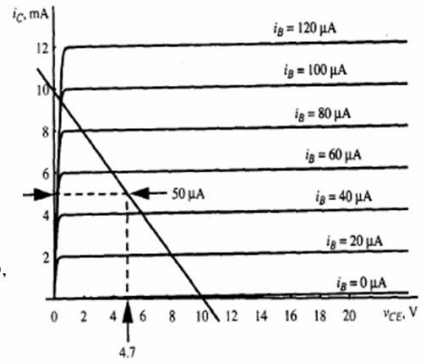
$I_B \le 0\,A$ *ise* ; *Kesmede* $I_B = I_C = 0A$ *Olur.*

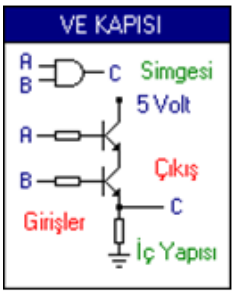
- *Transistörü kesimde tutmak için gereken max. gerilim v_{in} 0.7V dan 2.1V'a çıkar*
- *Transistör kesimden doyuma daha hızlı geçer*

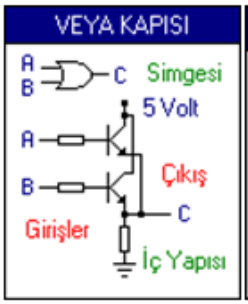
  (0.7V< v_{in}< 4V yerine 2.1V< v_{in}< 2.6V)

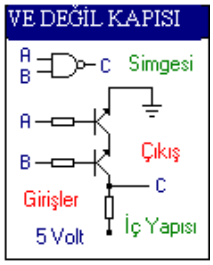




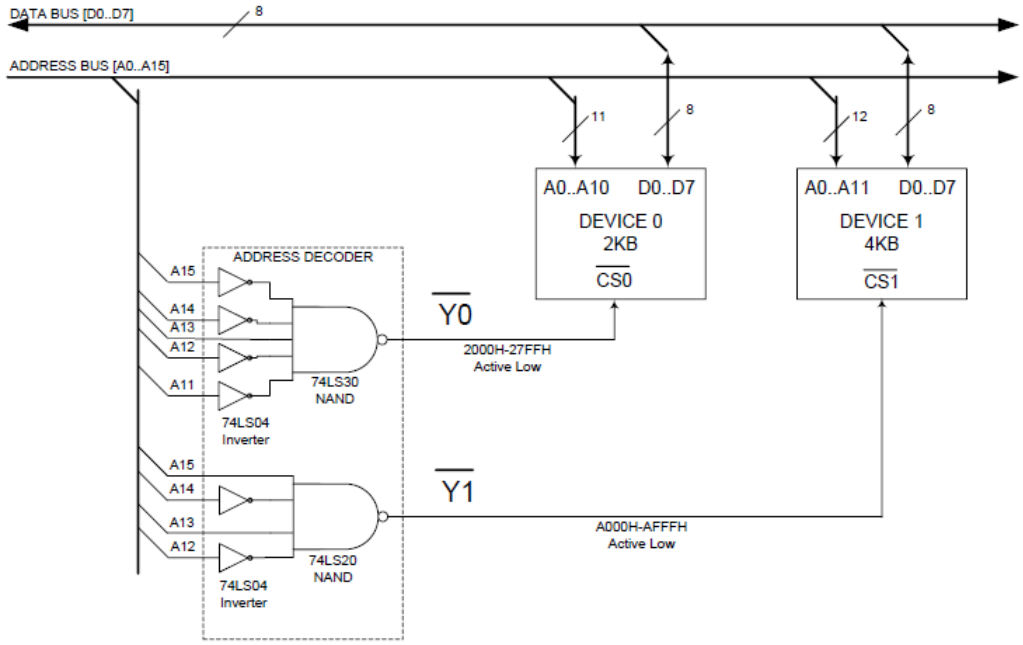








Örnek:



Block diagram of memory system

Device 0,

Chip select active low, NAND kapısı

CS0=A_{15} $A_{14}A_{13}A_{12}A_{11}$ =00100

Bellek erişi aralığı: A_{15} $A_{14}A_{13}A_{12}A_{11}A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1A_0$

0010 0000 0000 0000=2000h

0010 0111 1111 1111=27FFh

Bellek boyutu:

0010 0111 1111 1111+1 =0010 1000 0000 0000

Bellek boyutu= $2^{13} + 2^{11} – 2^{13} = 2^{11}$ byte= 2 x 2^{10} byte= 2KByte

Device 1,

Chip select active low, NAND kapısı

CS1=A_{15} $A_{14}A_{13}A_{12}$ =1010

Bellek erişim aralığı:

A_{15} $A_{14}A_{13}A_{12}A_{11}A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1A_0$

1010 0000 0000 0000=A000h

1010 1111 1111 1111=AFFFh

Bellek boyutu:

1010 1111 1111 1111+1 =1011 0000 0000 0000

Bellek boyutu= $2^{15} + 2^{13} + 2^{12} - 2^{15} - 2^{13} = 2^{12}$ byte= 4 x 2^{10} byte= 4KByte

Örnek:

8088 mikroişlemci ve dört adet bellekten oluşan sistemin mimarisini oluşturun.

Bellekler: U_0=1KByte, U_1=4Kbyte, U_2=2Kbyte, U_3=1Kbyte

8088 özellikleri:

Adres bus: 20 bit, (A_{19}, ... , A_0)

Data bus: 8 bit (D_7, ..., D_0)

Control bus: IOWR(0: Out, 1:Input), MWR (0: Write, 1: Read)

Memory mapping (Bellek eşleme):

U_0: 1Kbyte=2^{10}; Adres bus=10 bit (tel), (A_9, ... , A_0)

U_1: 4Kbyte=2^{12}; Adres bus=12 bit (tel), (A_{11}, ... , A_0)

U_2: 2Kbyte=2^{11}; Adres bus=11 bit (tel), (A_{10}, ... , A_0)

U_3: 1Kbyte=2^{10}; Adres bus=10 bit (tel), (A_9, ... , A_0)

Bellekler için CPU'dan çıkacak adres bus belirlenirken maksimum olan bellek elemanın adres bus'ı alınır: U_1: 4Kbyte=2^{12}; Adres bus=12 bit (tel), (A_{11}, ... , A_0)

Address decoding (Bellek seçme):

Toplam eleman sayısı=4<= 2^m

Bellek seçme için adres bus sayısı, m=2, Adres bus=A_{13}, A_{12}

A_{13}	A_{12}	M_0	M_1	M_2	M_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

CPU'dan çıkacak toplam adres bus sayısı=14<=20, (A_{13}, ... , A_0)

Örnek:

8086 mikroişlemci ve dört adet bellekten oluşan sistemin mimarisini oluşturun.

Bellekler: U_0=1KByte, U_1=2Kbyte, U_2=7Kbyte, U_3=4byte, U_4=12byte

8086 özellikleri:

Adres bus: 20 bit, (A_{19}, ... , A_0)

Data bus: 16 bit (D_{15}, ..., D_0)

Control bus: IOWR(0: Out, 1:Input), MWR (0: Write, 1: Read)

Memory mapping (Bellek eşleme):

U_0: 1Kbyte=2^{10}; adres bus=10 bit (tel), (A_9, ... , A_0)

U_1: 2Kbyte=2^{11}; adres bus=11 bit (tel), (A_{10}, ... , A_0)

U_2: 7Kbyte<=2^3 x 2^{10}=2^{13} ; Adres bus=13 bit (tel), (A_{12}, ... , A_0)

U_3: 4byte=2^2; adres bus=2 bit (tel), (A_1, A_0)

U_4: 12byte<=2^4; adres bus=4 bit (tel), (A_3, ... , A_0)

Bellekler için CPU'dan çıkacak adres bus belirlenirken maksimum olan bellek elemanın adres bus'ı alınır:

U_2: 7Kbyte<=2^3 x 2^{10}=2^{13} ; Adres bus=13 bit (tel), (A_{12}, ... , A_0)

Maksimum indeks=A_{12}

Address decoding (Bellek seçme):

Toplam eleman sayısı=5<= 2^m; m=3 alınır

Bellek seçme için adres bus sayısı bulunurken makimum indeksin devam alınır, m=3; Adres bus= A_{15}, A_{14}, A_{13}

A_{15}	A_{14}	A_{13}	M_0	M_1	M_2	M_3	M_4
0	0	0	1	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	0	0	1	0	0
0	1	1	0	0	0	1	0
1	0	0	0	0	0	0	1
1	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0

CPU'dan çıkacak toplam adres bus sayısı=16<=20, (A_{15}, ... , A_0)
$(3C76)_h$,hangi elemanın hangi bellek gözünü (8bit) gösterir?

A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0 = 0011 1100 0111 0110

Adres eşleme işleminden,
A_{15} A_{14} A_{13}=001, M_1 belleğine yazar.
A_{12} A_{11} A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0 =(1C76)h= $(0001\ 1100\ 0111\ 0110)_b = 2^{12} + 2^{11} + 2^{10} + 2^6 + 2^5 + 2^4 + 2^2 + 2^1 = 4096 + 2048 + 1024 + 64 + 32 + 16 + 4 + 2 = (7286)_D$

Not: M1 adres bus=2^{11} ; M1 beleğinin toplam göz sayısı 2048; $(7286)_D$ M1 belleğini içerisinde yoktur.

Örnek:
8088 mikroişlemci ve 12 adet 1Kbyte bellekten oluşan sistemin mimarisini oluşturun.

8088 özellikleri:
Adres bus: 20 bit, (A_{19}, ... , A_0)
Data bus: 8 bit (D_7, ..., D_0)
Control bus: IOWR(0: Out, 1:Input), MWR (0: Write, 1: Read)

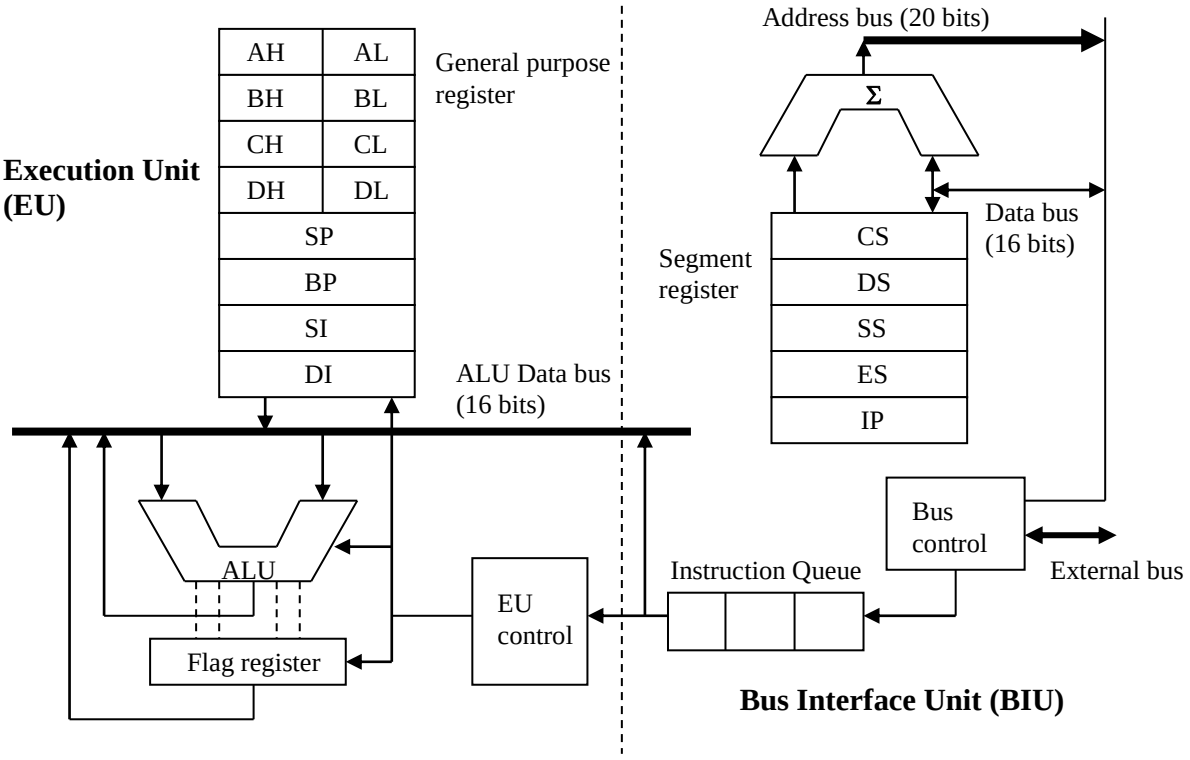
Memory mapping (Bellek eşleme):
U_0 , ... , U_{11}: 1Kbyte=2^{10}; Adres bus=10 bit (tel), (A_9, ... , A_0)

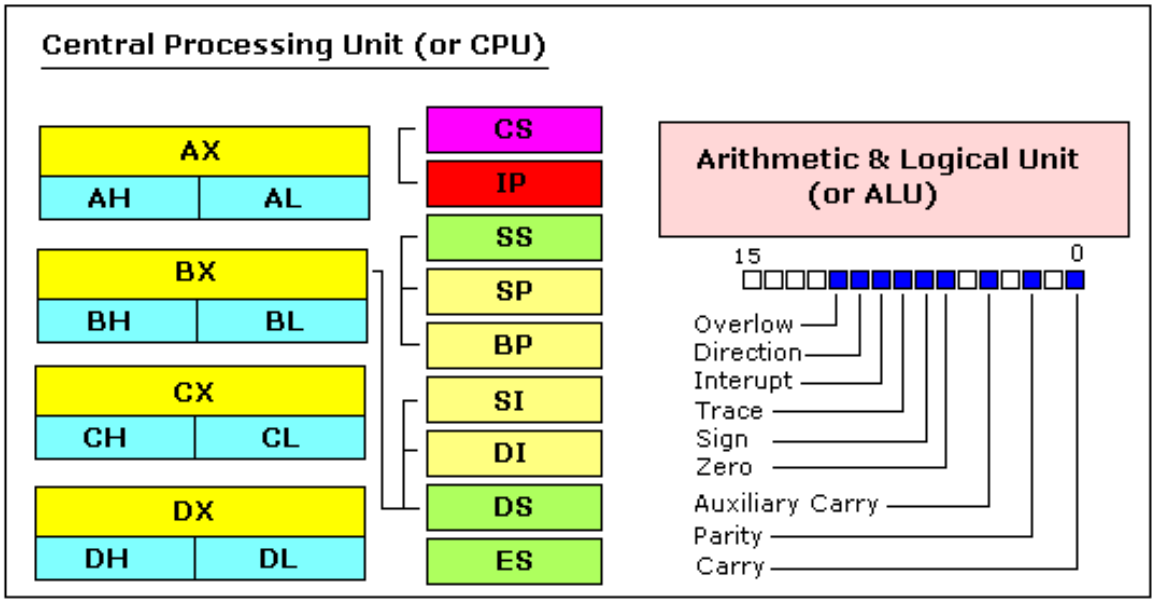
Bellekler için CPU'dan çıkacak adres bus belirlenirken maksimum olan bellek elemanın adres bus'ı alınır:
U_0 , ... , U_{11}: 1Kbyte=2^{10}; Adres bus=10 bit (tel), (A_9, ... , A_0)

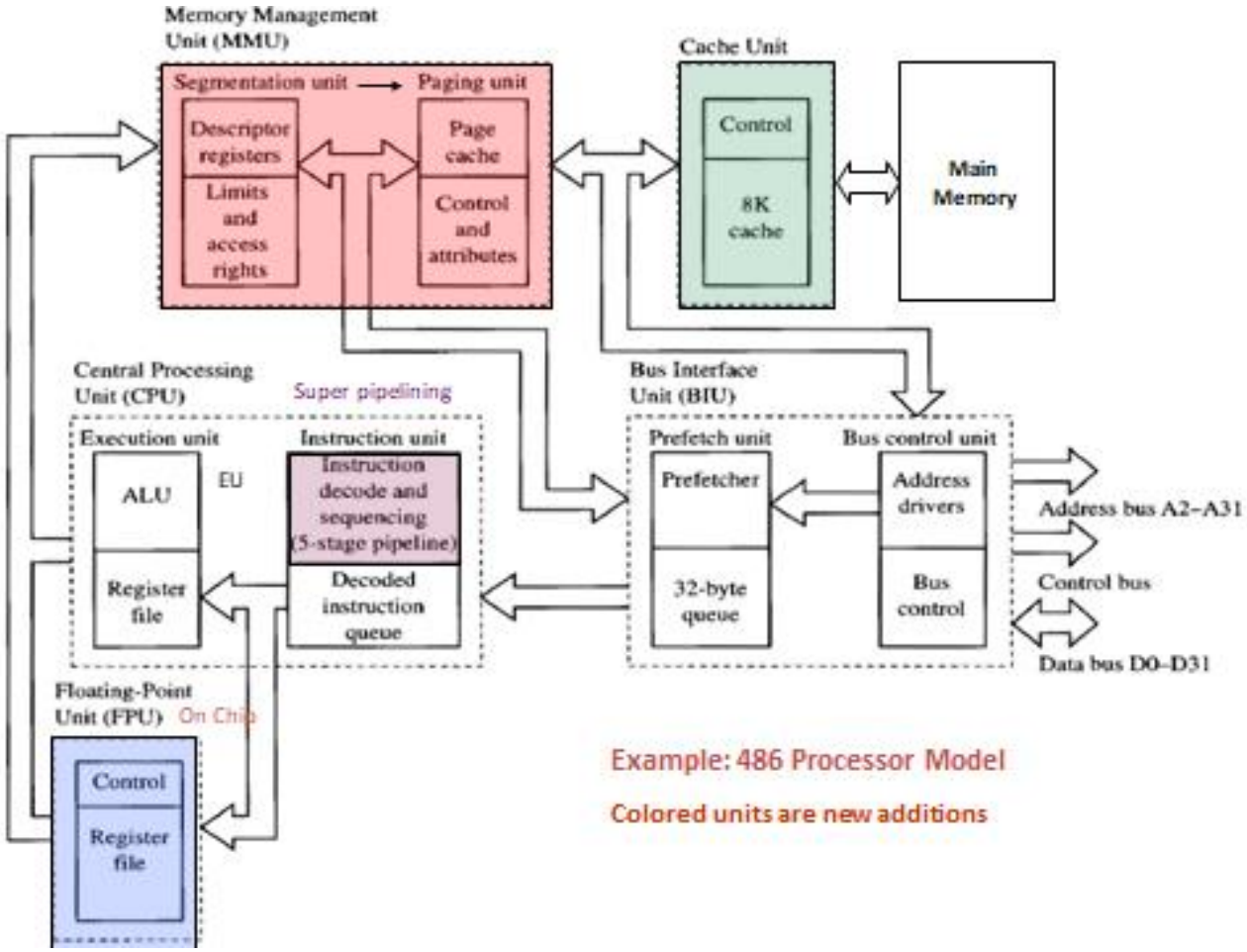
Address decoding (Bellek seçme):
Toplam eleman sayısı=12<= 2^m
Bellek seçme için adres bus sayısı, m=4, Adres bus=A_{13}, A_{12} , A_{11}, A_{10}

CPU'dan çıkacak toplam adres bus sayısı=14<=20, (A_{13}, ... , A_0)

2.4. CPU – Central Processing Unit





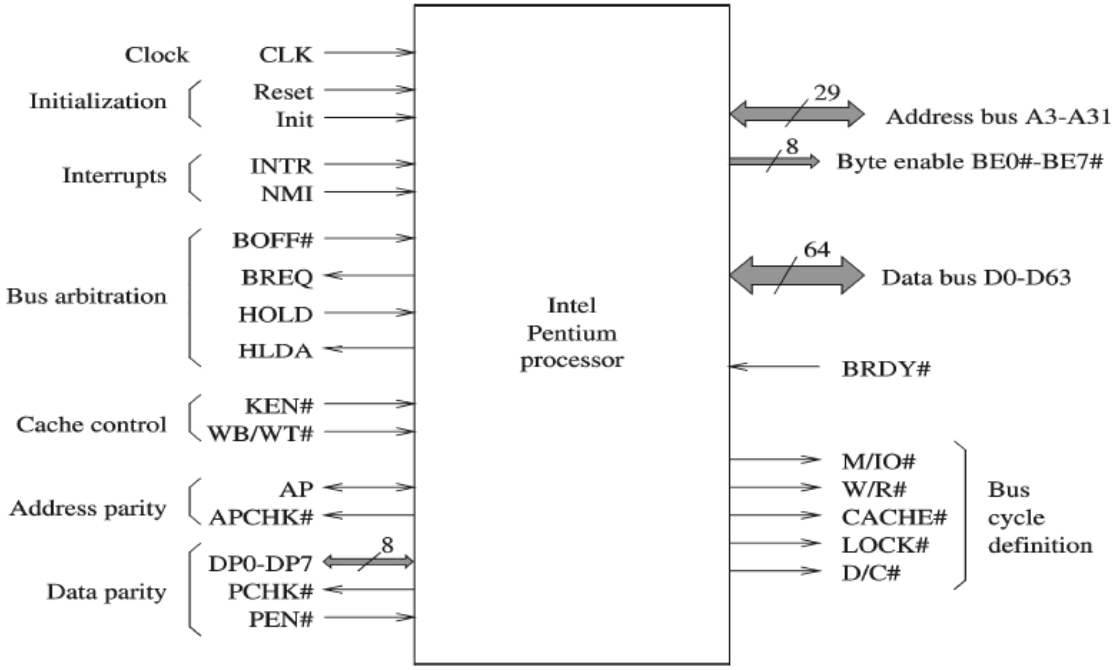


Example: 486 Processor Model

Colored units are new additions

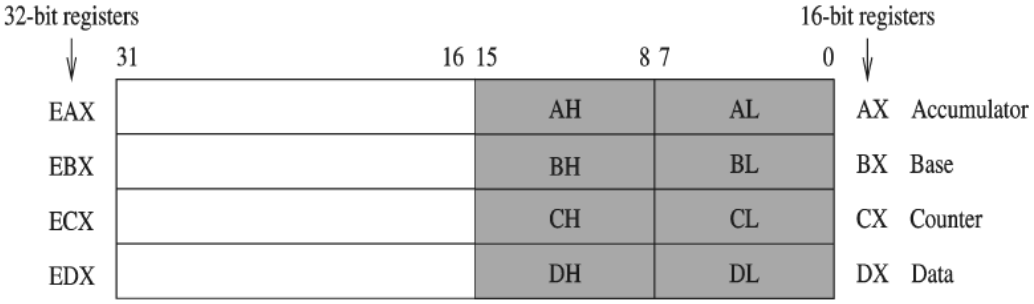
2.5. Pentium Processor

Pentium Processor



Pentium Registers

- Four 32-bit registers can be used as
  - Four 32-bit register (EAX, EBX, ECX, EDX)
  - Four 16-bit register (AX, BX, CX, DX)
  - Eight 8-bit register (AH, AL, BH, BL, CH, CL, DH, DL)
- Some registers have special use
  - ECX for count in loop instructions



2.6. x86 registers

16-bit

The original Intel **8086** and **8088** have fourteen 16-**bit** registers. Four of them (AX, BX, CX, DX) are general registers (although each may have an additional purpose; for example only CX can be used as a counter with the *loop* instruction). Each can be accessed as two separate bytes (thus BX's high byte can be accessed as BH and low byte as BL). Four segment registers (CS, DS, SS and ES) are used to form a memory address. There are two pointer registers. SP points to the bottom of the stack and BP which is used to point at some other place in the stack or the memory(Offset). Two registers (SI and DI) are for array indexing. The **FLAGS register** contains **flags** such as **carry flag**, **overflow flag** and **zero flag**. Finally, the instruction pointer (IP) points to the next instruction that will be fetched from memory and then executed.

32-bit

With the advent of the 32-bit 80386 processor, the 16-bit general-purpose registers, base registers, index registers, instruction pointer, and **FLAGS register**, but not the segment registers, were expanded to 32 bits. This is represented by prefixing an "**E**" (for **Extended**) to the register names in **x86 assembly language**. Thus, the AX register corresponds to the lowest 16 bits of the new 32-bit EAX register, SI corresponds to the lowest 16 bits of ESI, and so on. The general-purpose registers, base registers, and index registers can all be used as the base in addressing modes, and all of those registers except for the stack pointer can be used as the index in addressing modes. Two new segment registers (FS and GS) were added. With a greater number of registers, instructions and operands, the **machine code** format was expanded. To provide backward compatibility, segments with executable code can be marked as containing either 16-bit or 32-bit instructions. Special prefixes allow inclusion of 32-bit instructions in a 16-bit segment or vice versa.

64-bit

Starting with the AMD Opteron processor, the x86 in 64-bit long mode (as a subset of **x86-64** mode) extended the 32-bit registers in a way similar to what the 32-bit protected mode had done before (RAX, RBX, RCX, RDX, RSI, RDI, RBP, RSP, RFLAGS, RIP). However, 8 additional 64-bit general registers (R8, R9, ..., R15) were also introduced. The addressing modes were not dramatically changed from 32-bit mode, except that addressing was extended to 64 bits, physical addressing was now sign extended (so memory always added equally to the top and bottom of memory; note that this does not affect linear or virtual addressing), and other selector details were dramatically reduced.

x86 processors also include various special/miscellaneous registers such as **control registers** (CR0 through 4), **debug registers** (DR0 through 3, plus 6 and 7), **test registers** (TR4 through 7), descriptor registers (GDTR, LDTR, IDTR), and a task register (TR).

General purpose registers

CPU has 4 data registers, each register has its own name:

- **AX** - the accumulator register (divided into **AH / AL**).
- **BX** - the base address register (divided into **BH / BL**).
- **CX** - the count register (divided into **CH / CL**).
- **DX** - the data register (divided into **DH / DL**).

4 general purpose registers (AX, BX, CX, DX) are made of two separate 8 bit registers,

Index and pointer register

- **SI** - source index register.
- **DI** - destination index register.
- **BP** - base pointer.
- **SP** - stack pointer.

despite the name of a register, it's the programmer who determines the usage for each general purpose register. the main purpose of a register is to keep a number (variable). the size of the above registers is 16 bit, it's something like: **0011000000111001b** (in binary form), or **12345** in decimal (human) form.

AX= **0011000000111001b**, then AH=**00110000b** and AL=**00111001b**.

therefore, when you modify any of the 8 bit registers 16 bit register is also updated, and vice-versa. the same is for other 3 registers, "H" is for high and "L" is for low part.

because registers are located inside the CPU, they are much faster than memory. Accessing a memory location requires the use of a system bus, so it takes much longer. Accessing data in a register usually takes no time. therefore, you should try to keep variables in the registers. register sets are very small and most registers have special purposes which limit their use as variables, but they are still an excellent place to store temporary data of calculations.

Segment registers

Bellek başlangıç adresini gösteren saklayıcılardır.

- **CS** - points at the segment containing the current program.
- **DS** - generally points at segment where variables are defined.
- **ES** - extra segment register, it's up to a coder to define its usage.
- **SS** - points at the segment containing the stack.

although it is possible to store any data in the segment registers, this is never a good idea. the segment registers have a very special purpose - pointing at accessible blocks of memory.

segment registers work together with general purpose register to access any memory value. For example if we would like to access memory at the physical address **12345h** (hexadecimal), we should set the

DS = 1230h and **SI = 0045h**.

This is good, since this way we can access much more memory than with a single register that is limited to 16 bit values.

Although the main registers are "general-purpose" and can be used for anything, it was envisaged that they be used for the following purposes:

- AX/EAX/RAX: accumulator
- BX/EBX/RBX: base
- CX/ECX/RCX: counter
- DX/EDX/RDX: data/general
- SI/ESI/RSI: "source index" for **string** operations.
- DI/EDI/RDI: "destination index" for string operations.
- SP/ESP/RSP: stack pointer for top address of the stack.
- BP/EBP/RBP: stack base pointer for holding the address of the current **stack frame**.
- IP/EIP/RIP: instruction pointer. Holds the **program counter**, the current instruction address.

No particular purposes were envisioned for the other 8 registers available only in 64-bit mode. Some instructions compile and execute more efficiently when using these registers for their designed purpose. For example, using AL as an **accumulator** and adding an immediate byte value to it produces the efficient add to AL **opcode** of 04h, whilst using the BL register produces the generic and longer add to register opcode of 80C3h.

3. Memory and I/O ports

Group I : Addressing modes for register and immediate data

- Register Addressing
- Immediate Addressing

Group II : Addressing modes for memory data

- Direct Addressing
- Register Indirect Addressing
- Based Addressing
- Indexed Addressing
- Based Index Addressing
- String Addressing

Group III : Addressing modes for I/O ports

- Direct I/O port Addressing
- Indirect I/O port Addressing

Group IV : Relative Addressing mode

Group V : Implied Addressing mode

Memory: Store programs and data

Processor Memory

- Registers inside a microcomputer
- Store data and results temporarily
- No speed disparity
- Cost

Primary or Main Memory

- Storage area which can be directly accessed by microprocessor
- Store programs and data prior to execution
- Should not have speed disparity with processor Semi Conductor memories using CMOS technology
- ROM, EPROM, Static RAM, DRAM

Secondary Memory

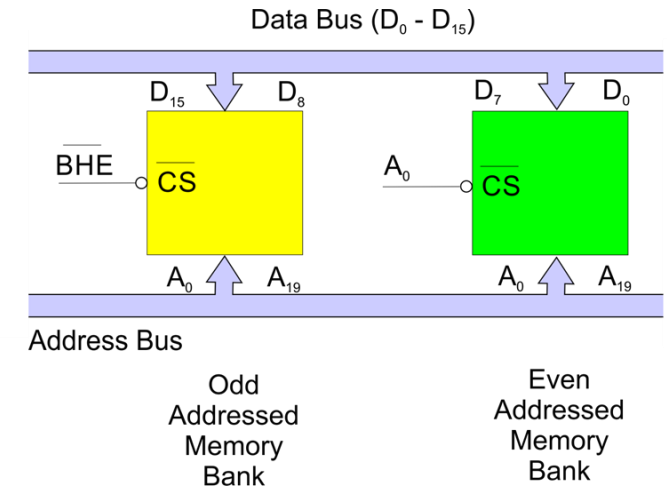
- Storage media comprising of slow devices such as magnetic tapes and disks
- Hold large data files and programs: Operating system, compilers, databases, permanent programs etc.

Memory mapping, I/O mapping

When memory mapping is used for I/O devices, full memory address space cannot be used for addressing memory.

3.1. Memory organization

Memory organization in 8086



Memory IC's : Byte oriented

8086 : 16-bit

Word : Stored by two consecutive memory locations; for LSB and MSB

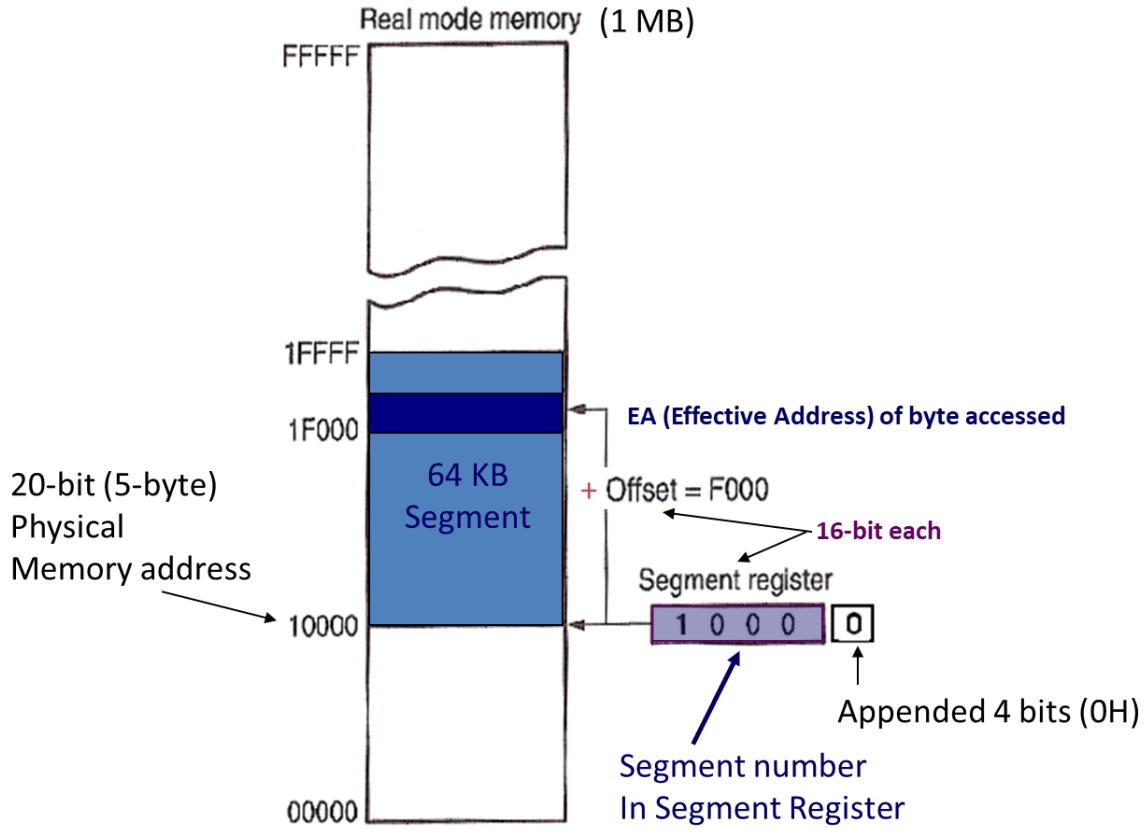
Address of word : Address of LSB

Bank 0 : A0 = 0 ise Even addressed memory bank

Bank 1 : $(\overline{BHE})$ = 0 ise Odd addressed memory bank

Addressing Modes : Memory Access

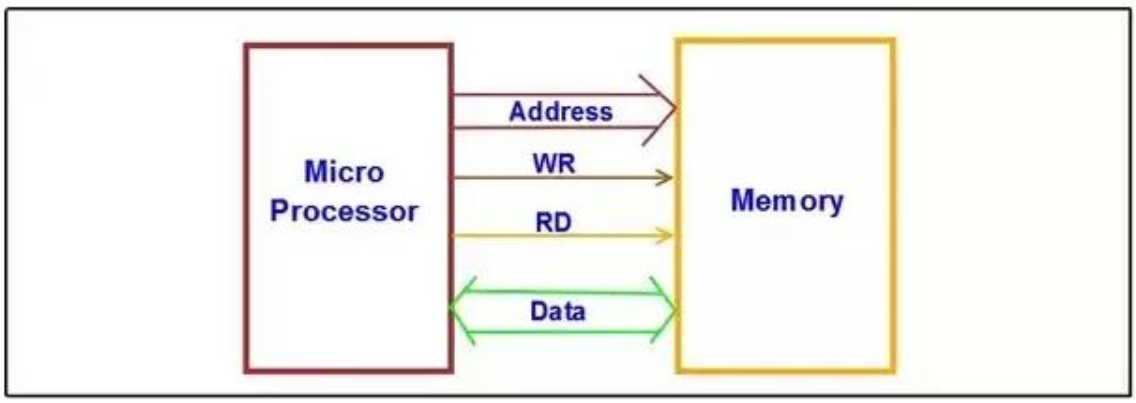
- 20 Address lines $\Rightarrow$ 8086 can address up to 2^{20} = 1M bytes of memory
- However, the largest register is only 16 bits
- Physical Address will have to be calculated Physical Address : Actual address of a byte in memory. i.e. the value which goes out onto the address bus.
- Memory Address represented in the form – Seg : Offset (Eg - 89AB:F012)
- Each time the processor wants to access memory, it takes the contents of a segment register, shifts it one hexadecimal place to the left (same as multiplying by 16_{10}), then add the required offset to form the 20- bit address



Segment	Offset (16-bit) 8080, 8086, 80286	Offset (32-bit) 80386 and above	Purpose
CS	IP	EIP	Program
SS	SP, BP	ESP, EBP	Stack
DS	BX, DI, SI, 8-bit or 16-bit #	EBX, EDI, ESI, EAX ECX, EDX, 8-bit or 32-bit #	Data
ES	DI, with string instructions	EDI, with string instructions	String destination

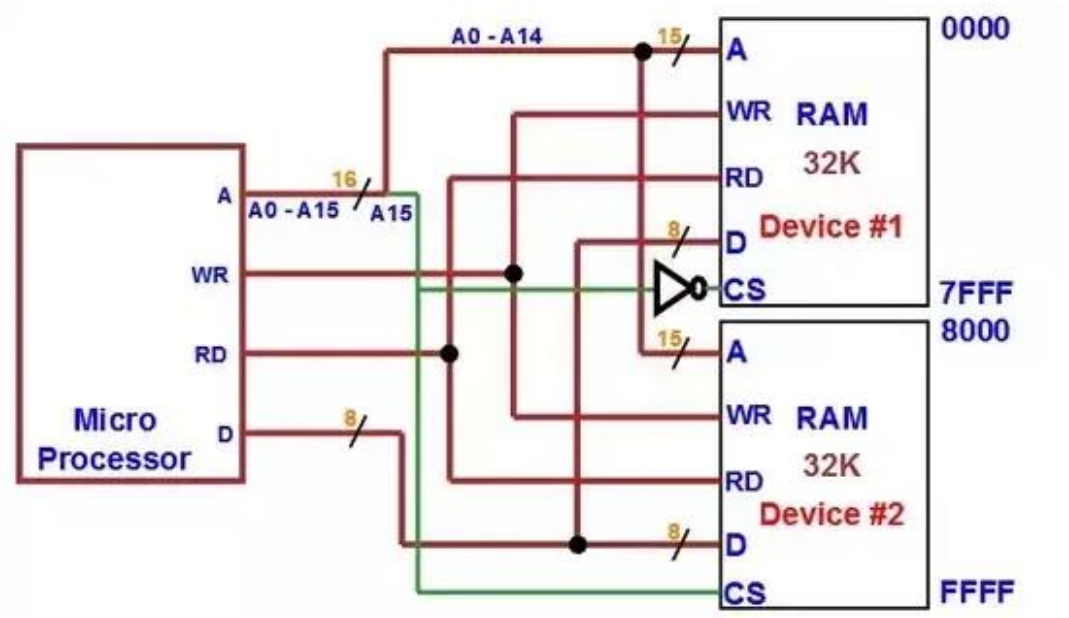
A process is a 32-bit process or a 64-bit process. For 32-bit processes, the portion of address space available for mapping consists of addresses in the range of 0x30000000-0xCFFFFFFF, for a total of 2.5G bytes of address space. The portion of address space available for mapping files consists of addresses in the rangesof 0x30000000-0xCFFFFFFF and 0xE0000000-0xEFFFFFFF for a total of 2.75G bytes of address space.

For 64-bit processes, two sets of address ranges with the process address space are available for mmap or shmat mappings. The first, consisting of the single range 0x07000000_00000000-0x07FFFFFF_FFFFFFFF, is available for both fixed-location and variable-location mappings. The second set of address ranges is available for fixed-location mappings only and consists of the ranges 0x30000000-0xCFFFFFFF, 0xE0000000-0xEFFFFFFF, and 0x10_00000000-0x06FFFFFF_FFFFFFFF. The last range of this set, consisting of 0x10_00000000-0x06FFFFFF_FFFFFFFF, is also made available to system loader to hold program text, data and heap, so only unused portions of the range are available for fixed-location mappings.



Address Lines	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Starting Address	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ending Address	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Processor	Memory		Data BUS	Mathematical Coprocessor
	Real	Virtual		
8086/8088	1 MB	-	16 / 8 b	8087
80286	16 MB	1 GB	16 b	80287
80386	4 GB	64 TB	32 b	80387
80486	4 GB	64 TB	32 b	integrated
PENTIUM	4 GB	64 TB	64 b	integrated
⋮	⋮	⋮	⋮	⋮
x86-32	4 GB	64 TB	64 b	integrated
⋮	⋮	⋮	⋮	⋮
x86-64	>1 TB	>64 TB	64 b	integrated



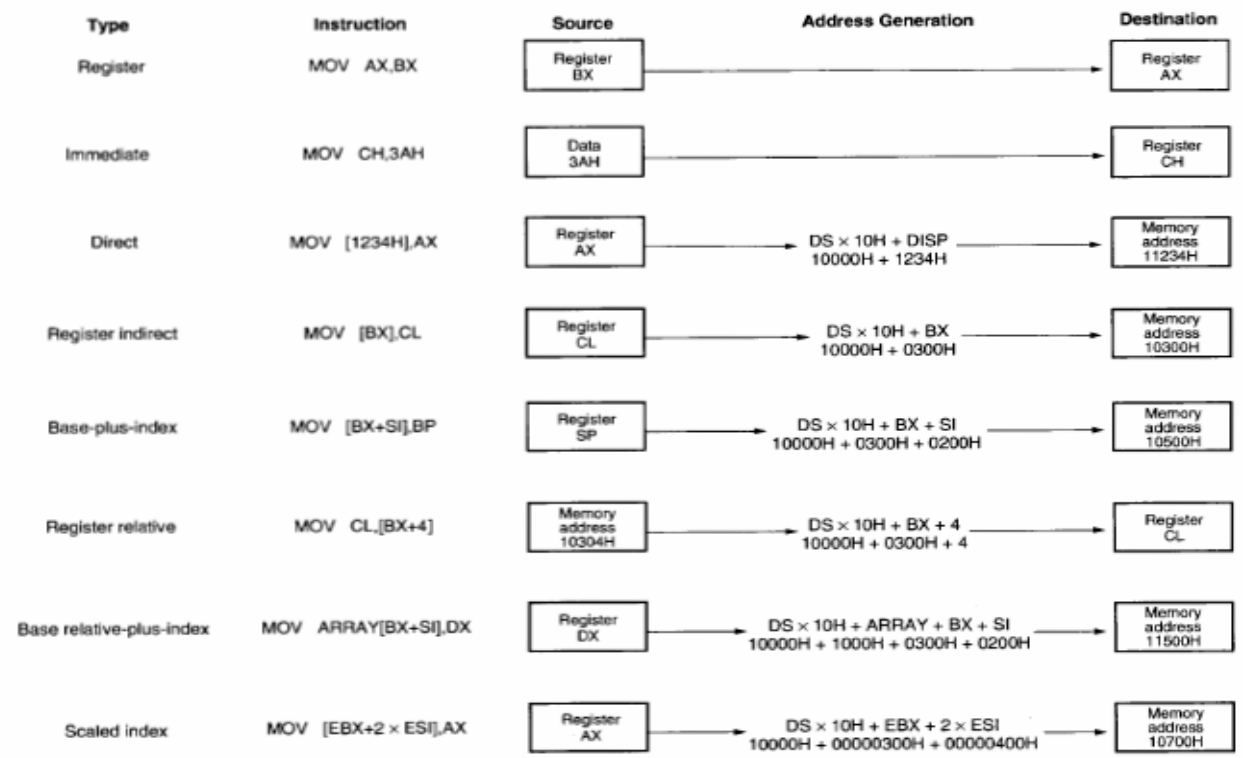
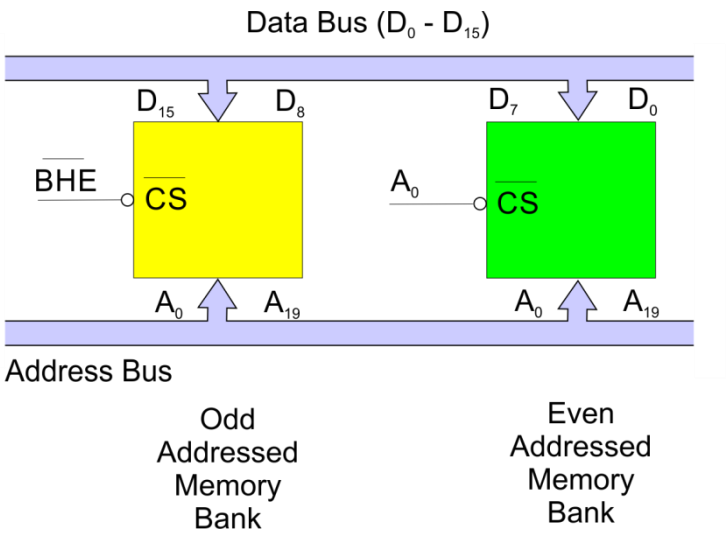


FIGURE 3–2 8086–Pentium 4 data-addressing modes.



	Operation		A_0	Data Lines Used
1	Read/ Write byte at an even address	1	0	$D_7 – D_0$
2	Read/ Write byte at an odd address	0	1	$D_{15} – D_8$
3	Read/ Write word at an even address	0	0	$D_{15} – D_0$
4	Read/ Write word at an odd address	0	1	$D_{15} – D_0$ in first operation byte from odd bank is transferred
		1	0	$D_7 – D_0$ in first operation byte from odd bank is transferred

3.2. Memory mapping

Available memory space = EPROM + RAM

Allot equal address space in odd and even bank for both EPROM and RAM

Can be implemented in two IC's (one for even and other for odd) or in multiple IC's

Interfacing SRAM and EPROM

Memory interface, Read from and write in to a set of semiconductor memory IC chip

EPROM, Read operations

RAM, Read and Write

In order to perform read/ write operations,

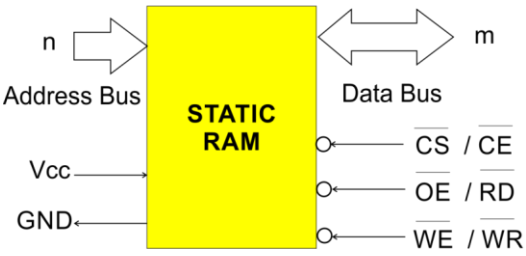
Memory access time: read / write time of the processor

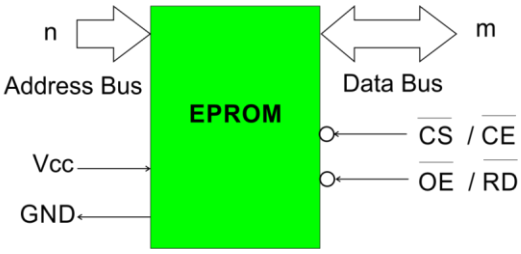
Chip Select (CS) signal has to be generated

Control signals for read / write operations

Allot address for each memory location

Interfacing SRAM and EPROM





Address pins:20, 2^{20} byte=1Mbyte,

Range of adress: 00000h to FFFFFh

Monitor Programs

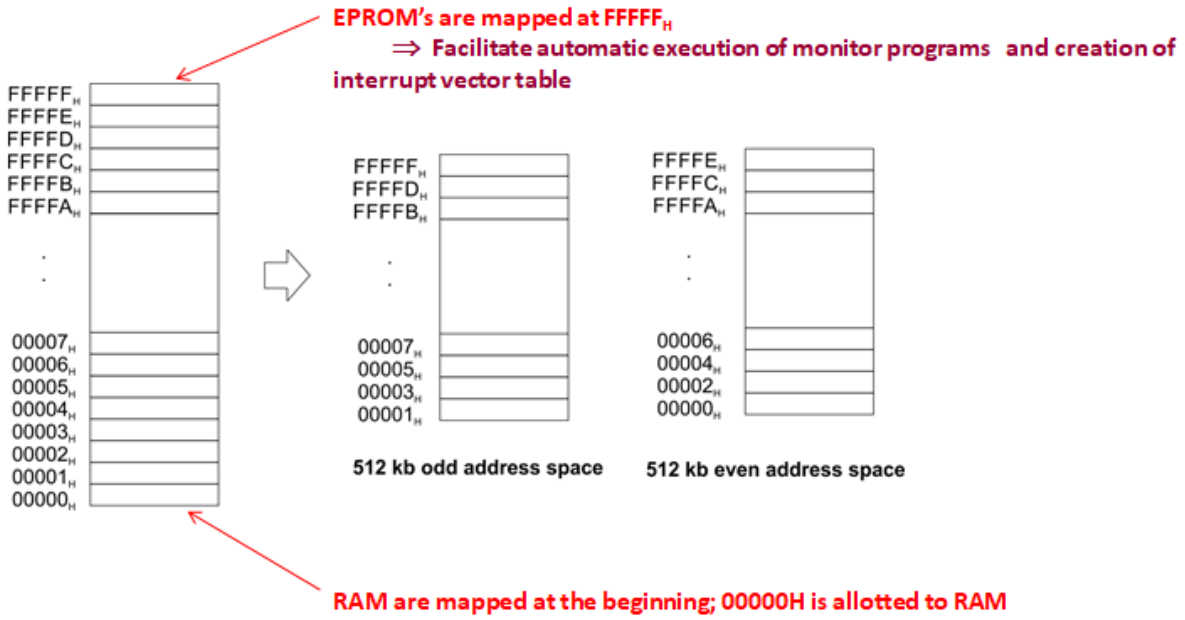
- Programing 8279 for keyboard scanning and display refreshing
- Programming peripheral IC's 8259, 8257, 8255, 8251, 8254 etc
- Initialization of stack
- Display a message on display (output)
- Initializing interrupt vector table

8279: Programmable keyboard/ display controller

8257: DMA controller

8259: Programmable interrupt controller

8255: Programmable peripheral interface



Soru:

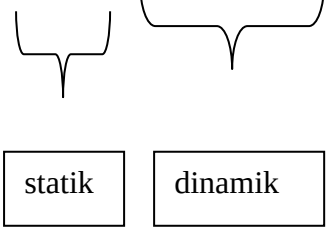
1001 0000 0000

.

.

.

1001 1111 1111



Bellek boyutunu bulunuz. 2^8=256

Endeks: 0, 1,2, ... , 255

Soru: Bellek endekslerini, boyutlarını bulunnuz.

	a15	a14	a13	a12	a11	a10	a9	a8	a7	a6	a5	a4	a3	a2	a1	a0	indeks		Kapasite
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	byte	
	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	2^12-1	byte	
	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2^12	byte	2^12
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2^14-1	byte	
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2^14	byte	4x2^12
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2^15-1	byte	
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2^15	byte	8x2^12
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2^16-1	byte	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			16x2^12

Bellek-1= 2^{12}byte=2^2 x 2^{10}byte=4 Kbyte

Endeks: (0000)h – (0FFF)h

Bellek-2=4 x 2^{12} - 2^{12}byte=3 x 2^2 x 2^{10}=12Kbyte

Endeks: (1000)h – (3FFF)h

Bellek-3=8 x 2^{12} – 4 x 2^{12} =4 x 2^{12} =16Kbyte

Endeks: (4000)h – (7FFF)h

Bellek-4=16 x 2^{12} – 8 x 2^{12} =8 x 2^{12} = 32Kbyte

Endeks: (8000)h – (FFFF)h

Toplam bellek kapasitesi: 4Kbyte + 12Kbyte + 16Kbyte +32Kbyte=64Kbyte

Soru:

CS: 16Kbyte

DS: 8Kbyte

SS:4Kbyte

ES: 4Kbyte

Bellek haritasını oluşturun ve başlangıçve bitiş endeksleri bulunuz.

CS bellek= 16Kyte= 2^4 x 2^{10} byte = 2^2 x 2^{12} byte=4 x 2^{12} byte

CS endeks=(0000)h –> (3FFF)h

DS bellek= 8Kbyte= 2^3 x 2^{10} byte=2 x 2^{12} byte

DS endeks=4 x 2^{12} byte -> 4 x 2^{12} + 2 x 2^{12} byte =4 x 2^{12} byte -> 6 x 2^{12} byte

DS endeks= (4000)h -> (5FFF)h

SS bellek= 4Kbyte= 2^2 x 2^{10} byte = 2^{12} byte

SS endeks=6 x 2^{12} byte -> 6 x 2^{12} byte + 2^{12} byte =7 x 2^{12} byte

SS endeks= (6000)h -> (6FFF)h

ES bellek= 4Kbyte= 2^2 x 2^{10} byte = 2^{12} byte

ES endeks=7 x 2^{12} byte -> 7 x 2^{12} byte + 2^{12} byte =8 x 2^{12} byte

ES endeks= (7000)h -> (7FFF)h

Soru bit toplam:

Elde	0	1	1		
	1	0	1	1	
+				1	
	1	1	0	0	

Elde	0	0	0	1	
	0	1	1	0	1
+					1
	0	1	1	1	0

	a15	a14	a13	a12	a11	a10	a9	a8	a7	a6	a5	a4	a3	a2	a1	a0	indeks		Kapasite
(0000)h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
																			16Kbyte
(3FFF)h	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
(4000)h	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4x2^12	byte	
																			24Kbyte-16Kbyte=8Kbyte
(5FFF)h	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1			
(6000)h	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6x2^12	byte	
																			28Kbyte-24Kbyte=4Kbyte
(6FFF)h	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1			
(7000)h	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	7x2^12	byte	
																			32Kbyte-28Kbyte=4Kbyte
(7FFF)h	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
(8000)h	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8x2^12	byte	

3.3. Interfacing I/O and peripheral devices

For communication between microprocessor and outside world.

Keyboards, CRT displays, Printers, Compact Discs etc.

Data transfer types

- Programmed I/ O, Data transfer is accomplished through an I/O port controlled by software
- Interrupt driven I/ O, I/O device interrupts the processor and initiate data transfer
- Direct memory access, Data transfer is achieved by bypassing the microprocessor

Input/OutputUnit(I/O)

Themainpurposeoftheinput/outputunitistoallowthemicroprocessortocommunicatedirectlywithpeopleorwithanothercomputerormachine.

The most familiar types of I/O equipment are,

a) input units: switches, keyboard, mouse, scanner, sensors,
microphone, photosensitive device.

b) output units: LEDs, LED display, video display, CRT, LCD, relays, solenoids, printer, speakers, motors.

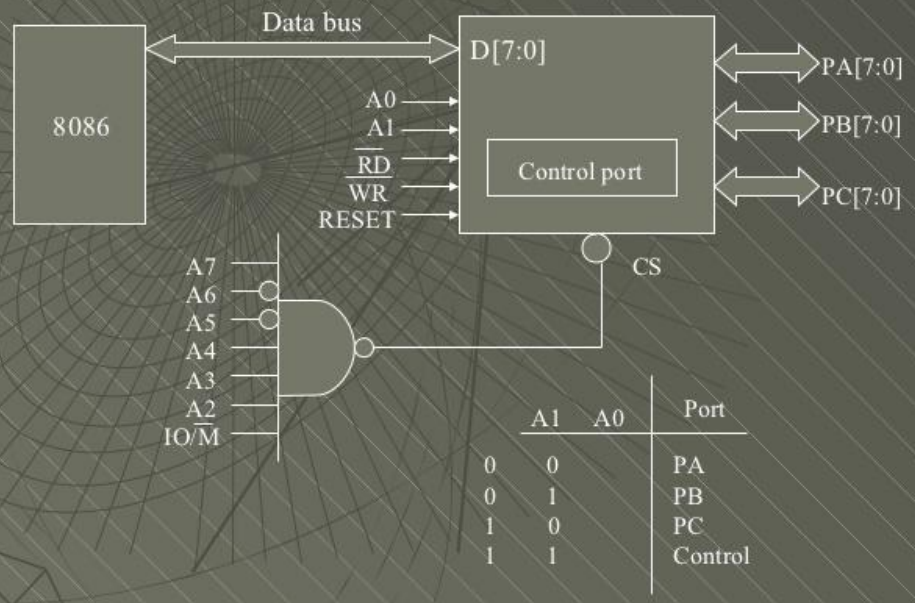
c) input/output units: disk drives, tape drives, USB-flash disk.

Typical microprocessor I/O equipment includes:

1.Analog-to-DigitalConverter(ADC),

2.Digital-to-AnalogConverter(DAC).

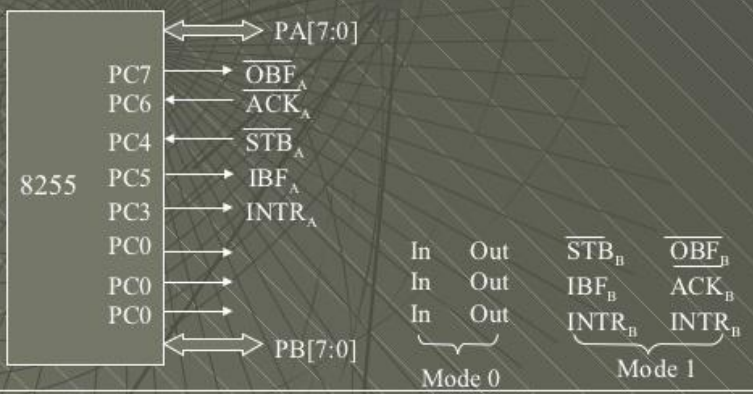
8255 Programmable Peripheral Interface



A1	A0	Port
0	0	PA
0	1	PB
1	0	PC
1	1	Control

Programming 8255

- Mode 2:
  - Port A is programmed to be bi-directional
  - Port C is for handshaking
  - Port B can be either input or output in mode 0 or mode 1

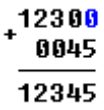


In	Out	$\overline{STB}_B$	$\overline{OBF}_B$
In	Out	IBF_B	$\overline{ACK}_B$
In	Out	$INTR_B$	$INTR_B$
Mode 0		Mode 1	

1. Can you design a decoder for an 8255 chip such that its base address is 40H?
2. Write the instructions that set 8255 into mode 0, port A as input, port B as output, PC0-PC3 as input, PC4-PC7 as output ?

3.4. Physical address

CPU makes a calculation of physical address by multiplying the segment register by 10h and adding general purpose register to it (1230h * 10h + 45h = 12345h):



the address formed with 2 registers is called an **effective address**. by default **BX, SI** and **DI** registers work with **DS** segment register; **BP** and **SP** work with **SS** segment register. other general purpose registers cannot form an effective address! also, although **BX** can form an effective address, **BH** and **BL** cannot.

special purpose registers

- **IP** - the instruction pointer.
- **flags register** - determines the current state of the microprocessor.

IP register always works together with **CS** segment register and it points to currently executing instruction.

flags register is modified automatically by CPU after mathematical operations, this allows to determine the type of the result, and to determine conditions to transfer control to other parts of the program.

generally you cannot access these registers directly, the way you can access AX and other general registers, but it is possible to change values of system registers using some tricks that you will learn a little bit later.

Addressing modes

Addressing modes for 16-bit x86 processors can be summarized by this formula:

$$\begin{Bmatrix} CS: \\ DS: \\ SS: \\ ES: \end{Bmatrix} \left[\begin{Bmatrix} BX \\ BP \end{Bmatrix} \right] + \left[\begin{Bmatrix} SI \\ DI \end{Bmatrix} \right] + [\text{displacement}]$$

Addressing modes for 32-bit address size on 32-bit or 64-bit x86 processors can be summarized by this formula:

$$\begin{Bmatrix} CS: \\ DS: \\ SS: \\ ES: \\ FS: \\ GS: \end{Bmatrix} \left[\begin{Bmatrix} EAX \\ EBX \\ ECX \\ EDX \\ ESP \\ EBP \\ ESI \\ EDI \end{Bmatrix} + \left[\begin{Bmatrix} EAX \\ EBX \\ ECX \\ EDX \\ EBP \\ ESI \\ EDI \end{Bmatrix} * \begin{Bmatrix} 1 \\ 2 \\ 4 \\ 8 \end{Bmatrix} \right] \right] + [\text{displacement}]$$

Addressing modes for 64-bit code on 64-bit x86 processors can be summarized by these formulas:

$$\begin{Bmatrix} : \\ FS: \\ GS: \end{Bmatrix} [\text{general register}] + \left[\text{general register} * \begin{Bmatrix} 1 \\ 2 \\ 4 \\ 8 \end{Bmatrix} \right] + [\text{displacement}]$$

And *RIP* + [displacement]

The 8086 had 64 KB of 8-bit (or alternatively 32 K-word of 16-bit) **I/O** space, and a 64 KB (one segment) **stack** in memory supported by **hardware**. Only words (2 bytes) can be pushed to the stack. The stack grows downwards (toward numerically lower addresses), its bottom being pointed by SS:SP. There are 256 **interrupts**, which can be invoked by both hardware and software. The interrupts can cascade, using the stack to store the **return address**.

3.5. Memory access

*The value in segment register (CS, DS, SS, ES) is called a **segment**, and the value in purpose register (BX, SI, DI, BP) is called an **offset**.* Displacement can be a değişken value or offset of a variable, or even both. if there are several values, assembler evaluates all values and calculates a single değişken value. Displacement can be inside or outside of the **[ ]** symbols, assembler generates the same machine code for both ways.

To access memory we can use these four registers: BX, SI, DI, BP, SP combining these registers inside **[ ]** symbols, we can get different memory locations. these combinations are supported (addressing modes):

[SI], [DI], [BX]
[BX + SI]
(DS-16bit, sağ tarafa 4bit 0 konur, 20 bit fiziksel adres elde edilir.)
Bx+ SI → gerçek fiziksel adres bulunur.

When DS contains value **b800h** and SI contains the value **10h** it can be also recorded as **b800:10**. [SI]=The physical address will be b8000h + 10h = b8010h.

d8 - stays for 8 bit signed değişken displacement (for example: 22, 55h, -1, etc...)
d16 - stays for 16 bit signed değişken displacement (for example: 300, 5517h, -259, etc...).
d16 (variable offset only)

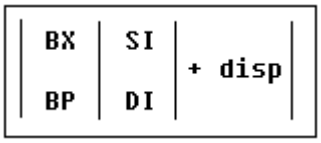
Displacement is a **signed** value, so it can be both positive or negative. Generally the compiler takes care about difference between **d8** and **d16**, and generates the required machine code.

for example, let's assume that **DS = 100**, **BX = 30**, **SI = 70**.
The following addressing mode: **[BX + SI] + 25** is calculated by processor to this physical address: **100 * 16 + 30 + 70 + 25 = 1725**.

[BX + DI], [BP + DI], [SI + d8]
[DI + d8], [BP + d8], [BX + d8]

By default **DS** segment register is used for all modes except those with **BP** register, for these **SS** segment register is used. There is an easy way to remember all those possible

combinations using this chart:



[BX + SI + d16], [BX + DI + d16], [BP + SI + d16], [BP + DI + d16]
[BX + SI + d8], [BX + DI + d8], [BP + SI + d8], [BP + DI + d8]
[SI + d16], [DI + d16], [BP + d16], [BX + d16]

You can form all valid combinations by taking only one item from each column or skipping the column by not taking anything from it. as you see **BX** and **BP** never go together. **SI** and **DI** also don't go together. here are an examples of a valid addressing modes: **[BX+5]** , **[BX+SI]** , **[DI+BX-4]**

if zero is added to a decimal number it is multiplied by 10, however **10h = 16**, so if zero is added to a hexadecimal value, it is multiplied by 16, for example: 7h = 7; 70h = $(112)_d$

In order to say the compiler about data type, these prefixes should be used: **byte ptr** - for byte. **word ptr** - for word (two bytes).

For example: byte ptr [BX] ; byte access. or word ptr [BX] ; word access.
assembler supports shorter prefixes as well:

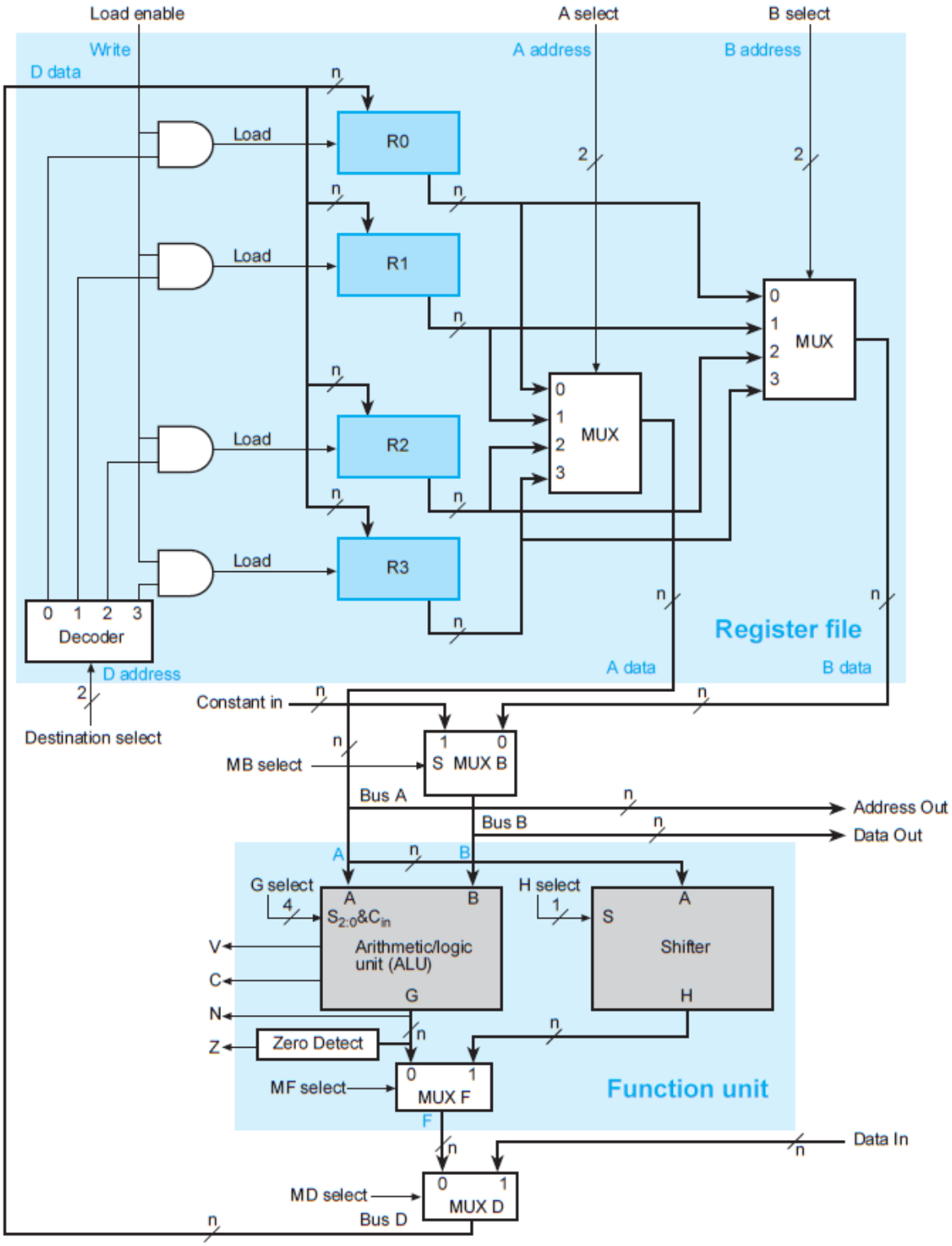
b. - for **byte ptr**
w. - for **word ptr**

in certain cases the assembler can calculate the data type automatically.

Stack is used by **CALL** instruction to keep return address for procedure, **RET** instruction gets this value from the stack and returns to that offset. Quite the same thing happens when **INT** instruction calls an interrupt, it stores in stack flag register, code segment and offset. **IRET** instruction is used to return from interrupt call. We can also use the stack to keep any other data, there are two instructions that work with the stack.

4. Mikroişlemci Mimari

Datapath ve Control unit bir mikroişlemcinin (CPU) iki parçasıdır. Datapath iki temel kısımdan oluşur, Register'lar ve Function Unit. Genel olarak bir Datapath'in block diagramı aşağıdaki şekildeki gibidir.



R1 ← R2 + R3 ifadesi, R2 ve R3 register'larındaki dataları toplayıp R1 register'ına yazma işlemini anlatır. R1 register'ına yükleme Load Enable girişinin aktif olması ile mümkündür. Load Enable girişi 1, Destination Select decoder'ının R1'e giden çıkşının da 1 olması gerekir.

A select'in bağlı bulunduğu Mux'dan R2'yi, B selectin bağlı bulunduğu Mux'dan R3'ü seçilir (A select=10, B select=11 olmalıdır.). Böylece A select'e bağlı olan Mux'dan çıkan R2 register'ındaki data yoluna devam ederek ALU'ya kadar ulaşır.

B select'e bağlı olan Mux'dan çıkan R3 register'ı ise MUXB'ye ulaşır. Bu Mux'da dışarıdan herhangi bir sabitle işlem yapılmayacağıiçin R3'ün yoluna devam etmesi için MB select=0 yapılır.

Şimdi R2 ve R3 register'larındaki dataların ikisi de ALU'ya ulaşmış olur. Bu durumda ne işlem yapılacağı seçilir. ALU, aritmetik ve logic işlemleri gerçekleyebilen bir yapıdır. Burada G select girişinden toplama işlemini yapan kod seçilir.

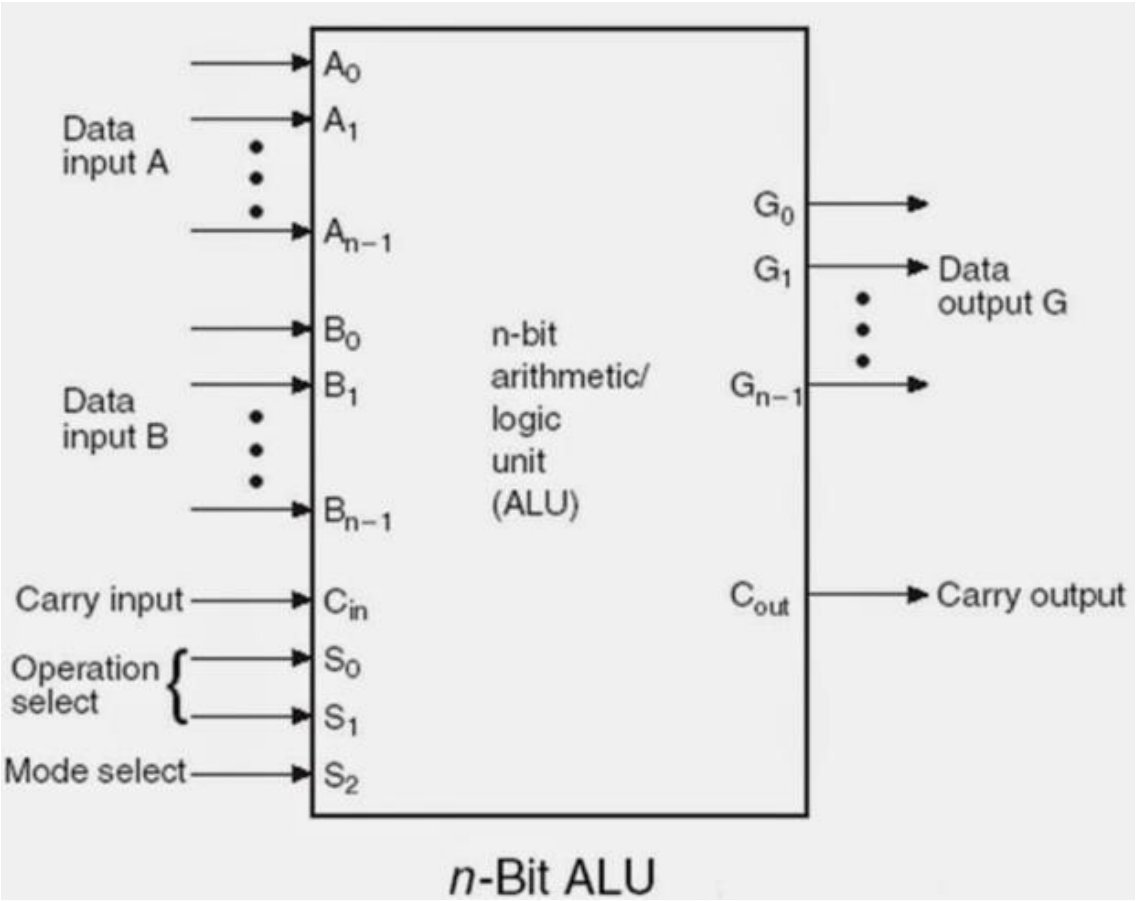
İşlem sonucu ALU'dan çıkış yaptıktan sonra MUXF'ye ulaşır. Burada MUXF'ye shifter'dan gelen bir giriş var. Eğer bir shift(kaydırma) işlemi olsaydı bu Mux'da 1 seçilir ancak ALU'dan gelen bilgi kullanılacağı için MF select =0 olmalıdır.

MUXF'den çıkış yapan bilgi MUXD'ye ulaşır. Bu Mux'un görevi ise dışarıdan bir datanın direk olarak bir register'a yazılması gerektiğinde 1 ucuna bağlı olan data in girişinden bilgiyi aldıktan sonra istenilen register'a yazar. Burada MUXF'den gelen data kullanılacağı için MD select=0 olmalıdır.

En sonunda MUXD'den çıkan data yolu takip ederek Load enable'ı aktif olan R1 register'ına yazılır.

4.1. Aritmetik Logic Unit(ALU)

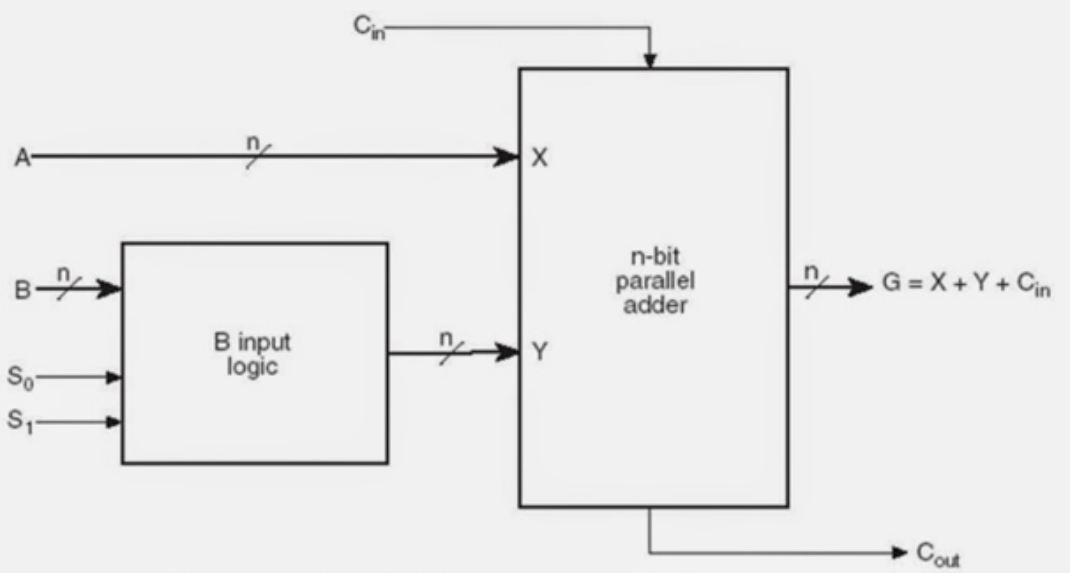
ALU aritmetik ve logic işlemlerin yapıldığı kombinasyonel mantık kapılarından oluşan devredir. ALU, Aritmetik ve Logic kısım olmak üzere iki kısma ayrılır. Herhangi bir clock pulsenin bulunmadığı kısımdır. ALU nun genel olarak yapısı şu şekildedir;



Şekilde görüldüğü üzere A'dan n-bit data, B'den n-bit data ve önceki işlemlerden gelebilecek bir elde değeri için bir Cin girişi var. S2 girişi işlemin aritmetik mi logic mi olacağına karar veren seçimdir. S1 ve S0 girişi ise aritmetik ya da mantıksal işlemin ne olacağını belirler. S1 ve S0 ile birlikte S2 girişi 0 olduğunda 8 adet farklı aritmetik işlem yaparak sonucu G çıkışından verir.

Aritmetik Devre:

Aritmetik devrenin temel parçasına Full Adder devresi denir. Paralel Adder'a giriş yapan datalar kontrol edilerek çeşitli aritmetik işlemleri gerçekleyebilmek mümkündür. Bir aritmetik devrenin block diagramı aşağıda verilmiştir.



Block Diagram of an Arithmetic Circuit

A'dan gelen datalar üzerinde değişiklik yapılmadan doğrudan Paralel Adder'a gider. Şimdi S0 ve S1'in durumlarına göre hangi işlemleri yapabileceğine bakalım.

Select		Input	$G = A + Y + C_{in}$	
S_1	S_0	Y	$C_{in} = 0$	$C_{in} = 1$
0	0	all 0's	$G = A$ (transfer)	$G = A + 1$ (increment)
0	1	B	$G = A + B$ (add)	$G = A + B + 1$
1	0	$\overline{B}$	$G = A + \overline{B}$	$G = A + \overline{B} + 1$ (subtract)
1	1	all 1's	$G = A - 1$ (decrement)	$G = A$ (transfer)

Burada Y'nin hangi S1,S0 değerinde bir seçim olacaktır.

- S1,S0=00 değerinde Y girişi 0 olur ve böylece Cin=0 ise G çıkışı A olacaktır. Yani sadece transfer işlemi yapılmış olur. Yine bu durumda Cin=1 ise bu kez G çıkışı A+1 değerini alacaktır.
- S1,S0=01 değerinde Y çıkışına B datası doğrudan aktarılır. Cin=0 ise G=A+B, Cin=1 ise G=A+B+1 olacaktır.
- S1,S0=10 değerinde Y çıkışına B datası terslenerek aktarılır. Cin=0 ise G=A+B' .Cin=1 ise G=A+B'+1 dolayısıyla bu da G= A-B değerine eşit olacaktır.

- S1,S0=11 değerinde Y çıkışı 1 olur. Buradaki 1'in kaç bit olacağı merak edilebilir. A ve B biti kaç bitse çıkıştaki 1 sayısı da o kadar bittir. A'nın tüm bitlerini 1 ile topladığınızda A daki datanın eğer Cin=0'sa 1 eksiğini aldığınızı farkedeceksiniz. Eğer Cin=1 ise mantıksal olarak A-1+1=A olacaktır.

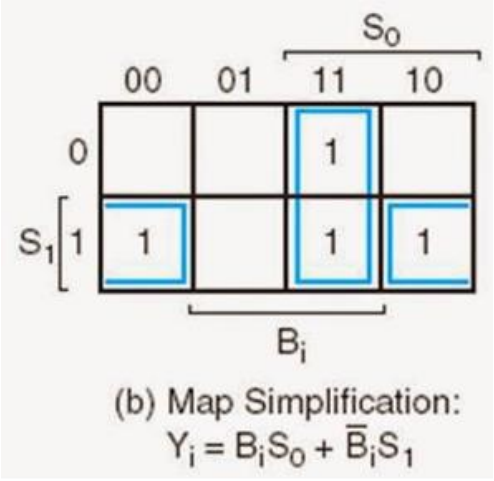
Not: Burada işimize yarayacak olan en kullanışlı sonuçlar;

- G=A (transfer)
- G=A+1 (increment)
- G=A+B (toplama)
- G=A+B'+1=A-B (çıkarma)
- G=A-1 (decrement)

Buradaki ifadeler 4 to 1 MUX kullanarak yapılabilir. Logic kapılarla çözüm elde edilebilir. Doğruluk tablosundaki Bi, B datasının herhangi bir bitidir.

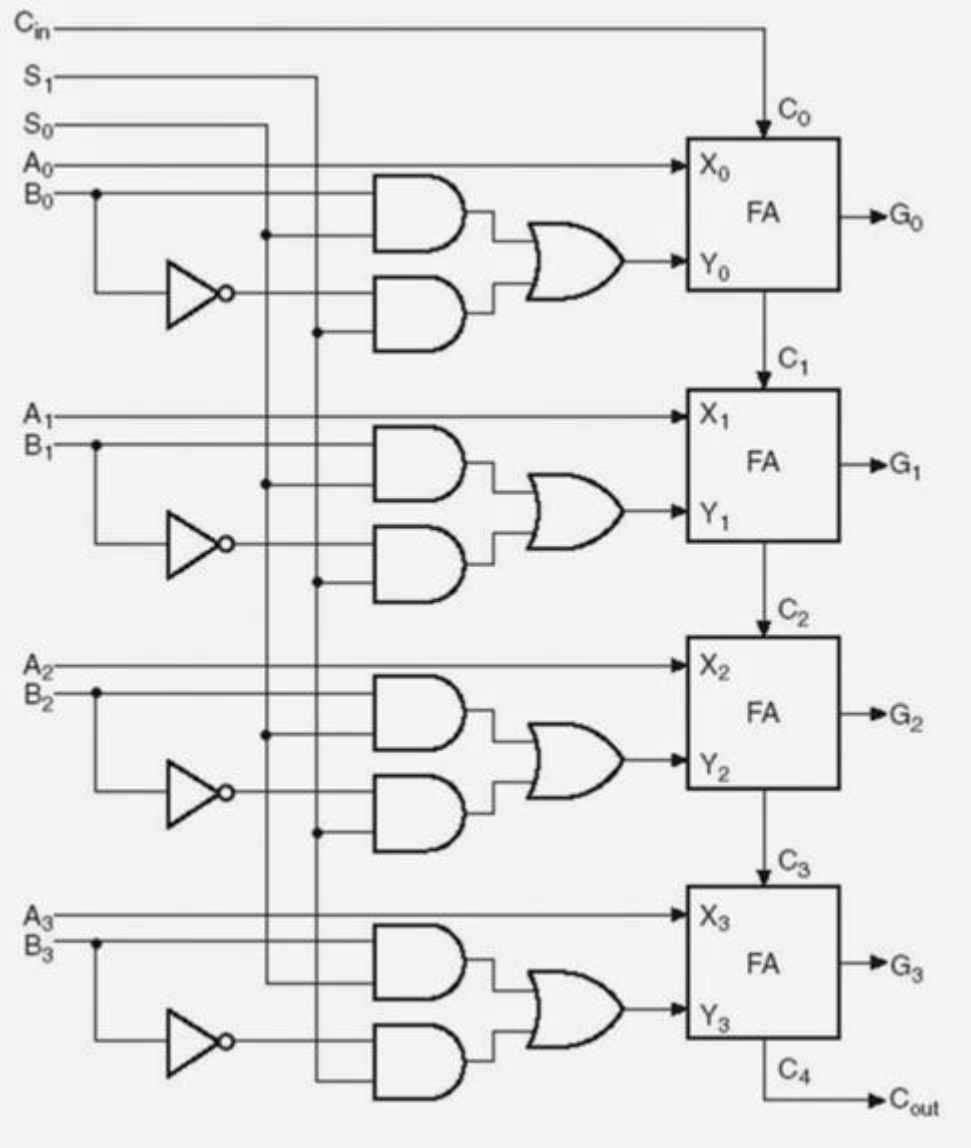
Inputs			Output	
S_1	S_0	B_i	Y_i	
0	0	0	0	$Y_i = 0$
0	0	1	0	
0	1	0	0	$Y_i = B_i$
0	1	1	1	
1	0	0	1	$Y_i = \overline{B}_i$
1	0	1	0	
1	1	0	1	$Y_i = 1$
1	1	1	1	

Bu tablodan Yi çıkışı için mantıkal ifade çıkarmak istersek Karnough diyagramı kullanılır.



(b) Map Simplification:
$Y_i = B_i S_0 + \overline{B}_i S_1$

Gerekli sadeleştirme yapıldıktan sonra Yi= BiS0+Bi'S1 bulunur. B nin 4 bitlik bir datadan oluştuğunu varsayarsak logic devre şu halde olacaktır.



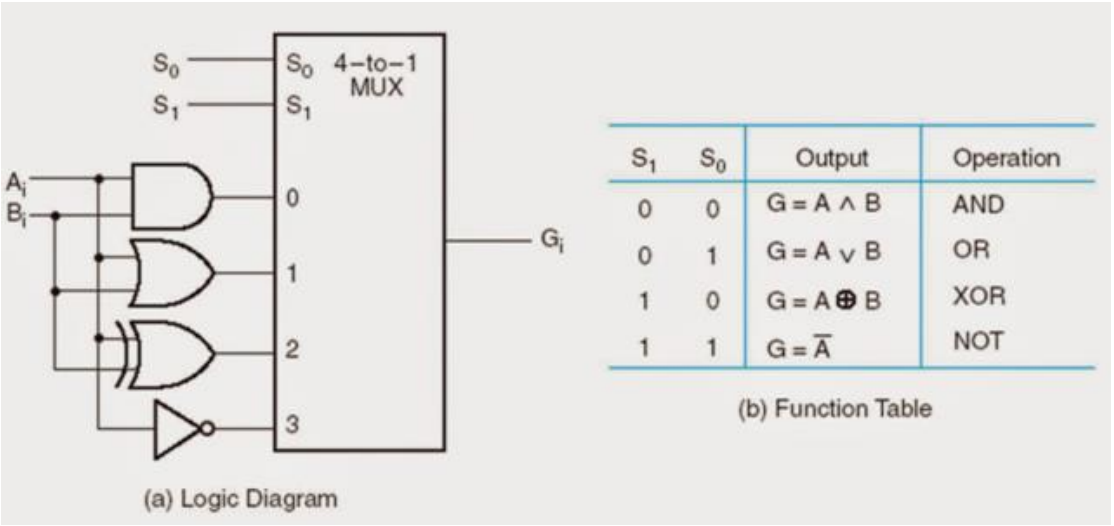
Burada bit sayısı istenilen kadar arttırılabilir. Böylece ALU nun aritmetik kısmını tasarlanmış olunur.

Logic devre:

Bu devrede kullanabilecek bir çok logic ifade olabilir. Ancak çok sık kulanılanlardan devreyi oluşturulması gerekir. Genelde en sık kullanılan logic ifadeler,

- AND
- OR
- XOR
- NOT

Bu logic ifadeleri seçime bağlı şekilde çıkışa verebilecek bir devre tasarlanılması gerekiyor. Bunun için 4 to 1 MUX kullanılır. Aşağıda bu devrenin gerçeklenmiş şekli bulunmaktadır.



(a) Logic Diagram

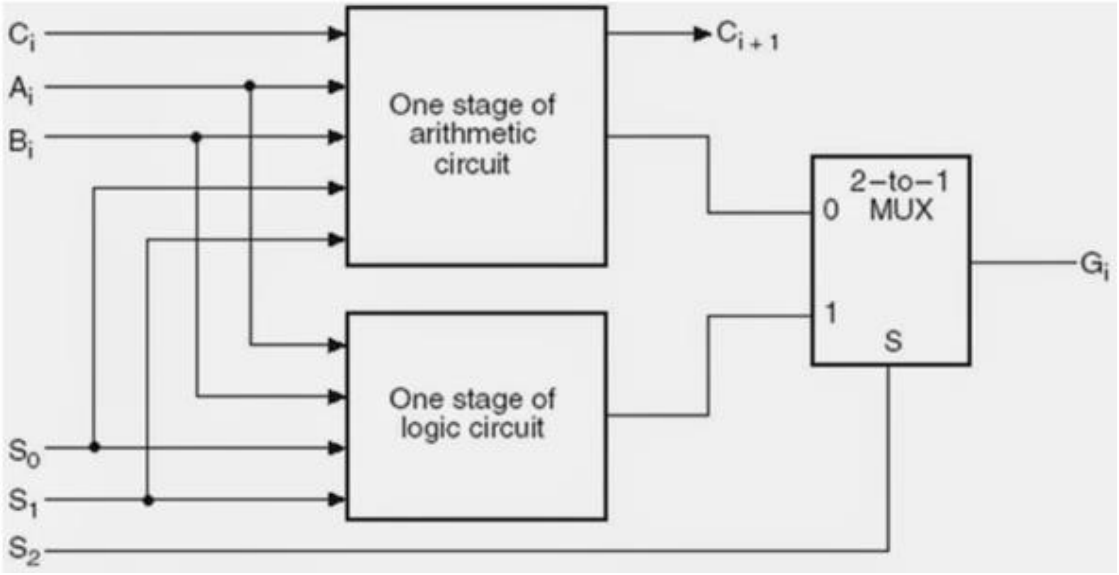
S_1	S_0	Output	Operation
0	0	$G = A \wedge B$	AND
0	1	$G = A \vee B$	OR
1	0	$G = A \oplus B$	XOR
1	1	$G = \overline{A}$	NOT

(b) Function Table

Logic devre de tamamlandığında aritmetik devre ile birleştirip ALU oluşturulur.

Aritmetik Logic unit:

Aritmetik devre ile logic devre birleştirilirken S2 girişi yardımıyla istenildiğinde logic devre, istenildiğinde aritmetik devre seçimi bir mux yardımı ile yapılabilir.



Devrede görüldüğü gibi S2=0 olduğunda aritmetik, S2=1 olduğunda logic işlem aktif olacaktır. Burada dikkat edilmesi gereken önemli bir nokta aritmetik ve logic devrenin S0 ve S1 girişlerinin ortak olarak kullanılmasıdır. S2 seçimine göre yalnız bir devrede aktif olabilceği için bunun hiç bir zararı yoktur. Hatta devreyi basitleştirir. Sonuç olarak ALU' nun içerisinde 8 aritmetik 4 logic işlem yapabilme imkanı bulunmaktadır. Yukarıda verile değerlere göre oluşturulan ALU'nun doğruluk tablosu aşağıdaki gibi olacaktır.

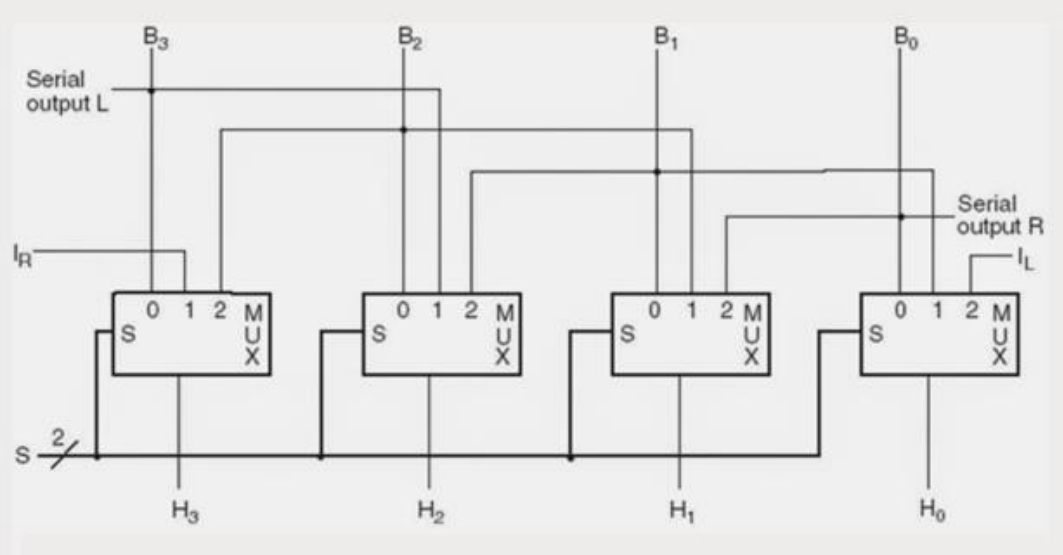
Operation Select					
S_2	S_1	S_0	C_{in}	Operation	Function
0	0	0	0	$G = A$	Transfer A
0	0	0	1	$G = A + 1$	Increment A
0	0	1	0	$G = A + B$	Addition
0	0	1	1	$G = A + \underline{B} + 1$	Add with carry input of 1
0	1	0	0	$G = A + \underline{B}$	A plus 1's complement of B
0	1	0	1	$G = A + \overline{B} + 1$	Subtraction
0	1	1	0	$G = A - 1$	Decrement A
0	1	1	1	$G = A$	Transfer A
1	0	0	X	$G = A \wedge B$	AND
1	0	1	X	$G = A \vee B$	OR
1	1	0	X	$G = \underline{A} \oplus B$	XOR
1	1	1	X	$G = A$	NOT (1's complement)

Şimdi de datapath içindeki ALU'nun sağ tarafında bulunan shifter kısmı tasarlanacak.

4.2. Mantıksal Devreler

Shifter:

Datapath devresinde Shifter'a sadece B select kısmından gelen bilgi giriş yapar. Bu nedenle iki farklı datayı bu datapath içinde kaydırma(shift) işlemine tabi tutulamaz. Şimdi 4 bitlik bir shifter yapısını inceleyelim,



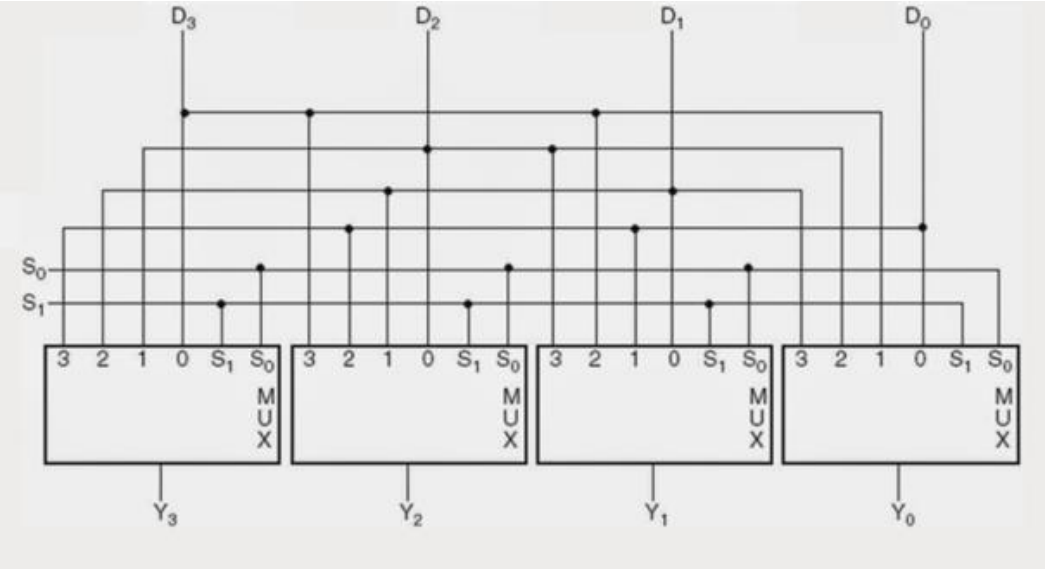
Şekilde görüleceği gibi Shifter, Clock pulse gereği duyulmadan Mux'lar ile kurulmuştur. Bundan dolayı datapath içinde bir shift işlemi yapıp Bus H yoluna(datapath resminde görülen) iletip yükleme yapılacak registera iletmek için bir Clock pulse yeterli olcaktır. şimdi şekildeki 2 bitlik S girişine göre hangi shift işlemlerinin yapılacağını tayin edelim. Burada belirtilmeli ki, hangi 2 bitlik S değerine göre hangi shift işleminin yapılacağı keyfi bir biçimde sıralanır.

- S=00 iken hiç kaydırma yapmadan Bus H yoluna direk aktarılır ki bunun işimize ne kadar yarayacağı tartışma konusu olabilir.
- S=01 iken Right-shift işlemi yapılır. Örneğin B registerındaki data 1010 olsun bunun Bus H ye aktarılmış şekli Ir101 olacaktır. Burada Ir binary bir değerdir eğer Ir=0 ise 0101 ; eğer Ir=1 ise 1101 şeklinde Bus H'ya aktarılacaktır
- S=10 iken Left-shift işlemi yapılır. Yine benzer şekilde bu defa IL değerine bağlı olmak üzere bir bit sola kaydırılarak data Bus H'a aktarılır.

Bu Shifter tipinde bir clock evresinde yalnızca bir bit kaydırma işlemi yapılabilir. Datapath uygulamalarında 1 clock evresinde 1 pozisyon kaydırmadan daha fazlasına ihtiyaç olduğunda Barrel shifter kullanılır.

Barrel Shifter:

4 bitlik Barrel shifter yapısı:



Devremiz yine kombinasyonel bir devre olduğu için istenilen registera datanın kaydırılıp yükleme işlemi yapılması sadece bir clock pulse süresinde gerçekleşecektir.

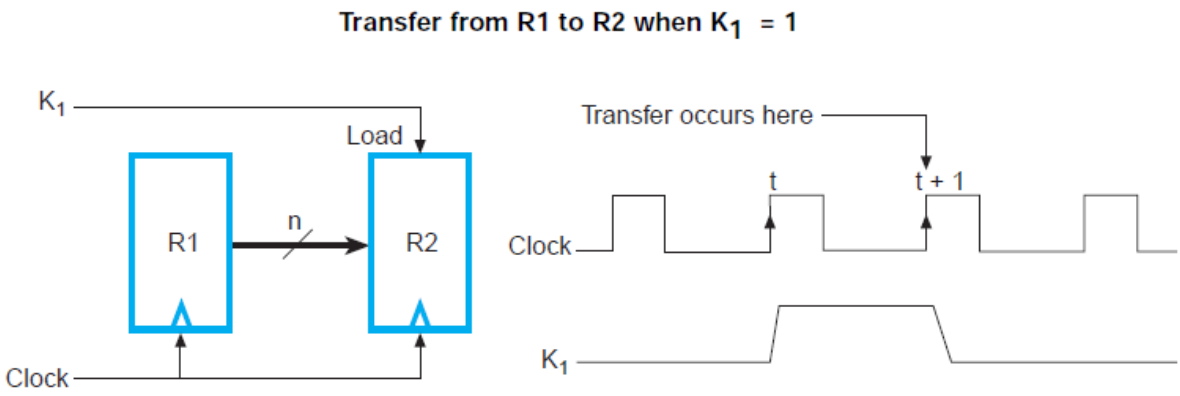
Barrel shifter ın doğruluk tablosu:

Select		Output				
S_1	S_0	Y_3	Y_2	Y_1	Y_0	Operation
0	0	D_3	D_2	D_1	D_0	No rotation
0	1	D_2	D_1	D_0	D_3	Rotate one position
1	0	D_1	D_0	D_3	D_2	Rotate two positions
1	1	D_0	D_3	D_2	D_1	Rotate three positions

S1,S0=10 durumunda ve S1,S0=11 durumunda kaydırma işlemleri tek seferde yapılmaktadır. Önceki Shifter'da bunun yapılması için bir bit kaydırıldıktan sonra tekrar shiftera sokup bir bit daha ve ihtiyaç varsa bir daha bir daha shifter'a sokulması gerekirdi. Ancak Barrel shifter'ın avantajı bu işlemleri tek seferde yapabiliyor olması. Ancak bunun da dezavantajı var. Şekli incelersek aslında tam bir shift işlemi yapılmıyor. Önceki shifter'da kaydırılan datanın boş kalan yerine 0'mı 1 mi kayacağımız IR ve IL girişleri ile belirlenebiliyordu. Barrel shifter'da yapılan işlem ise aslında döndürme(rotation) işlemi örneğin sağa doğru 1 pozisyon döndürme işlemi yaptığımızda en sağda kaybolan bit data kaydırıldıktan sonra en soluna yazılarak döndürme işlemi yapılıyor. Yine iki ve üç pozisyan kaydırma işlemi de buna benzer şekilde çalışıyor. Şekildeki yollar incelenirse daha rahat kavranabilir.

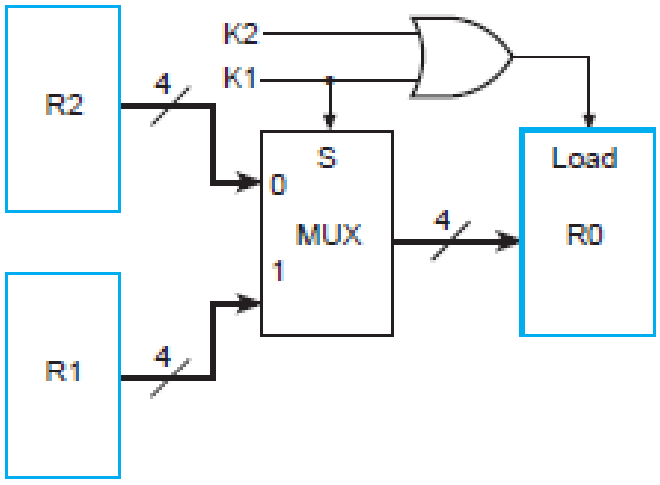
Not: İki shifter tipinde de kaç bit varsa o bit sayısında Mux'a ihtiyaç duyduğumuza dikkat edin.

R1 Register'ından R2 Register'ına Transfer:

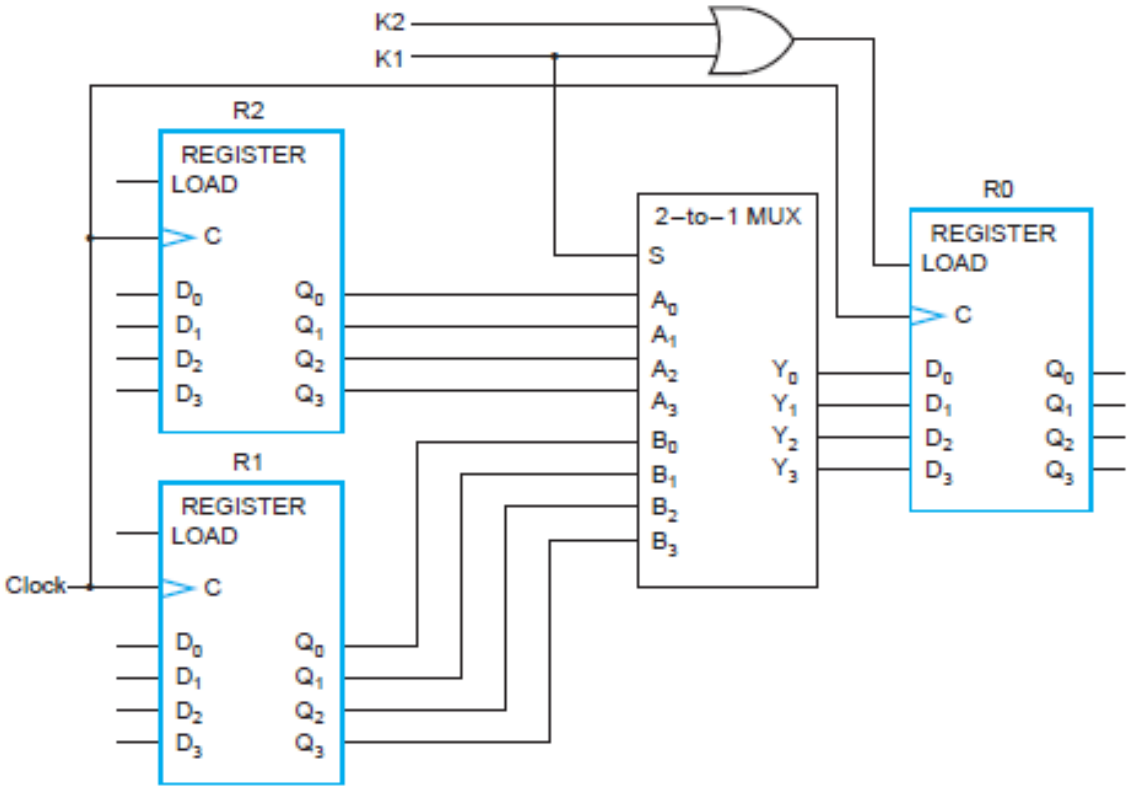


İki Register Arasında Seçim Yapmak İçin Çoklayıcıların Kullanımı

Blok Diyagram:

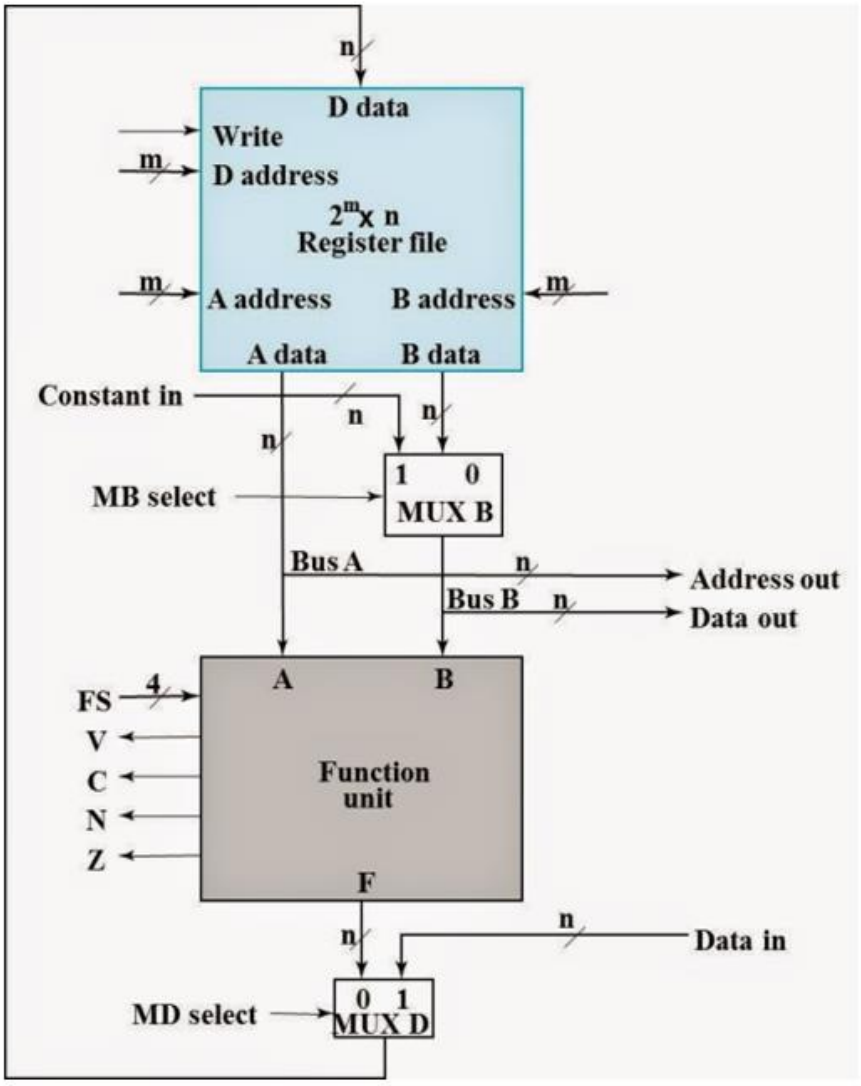


Ayrıntılı mantık devresi:



4.3. Datapath'in İrdelenmesi

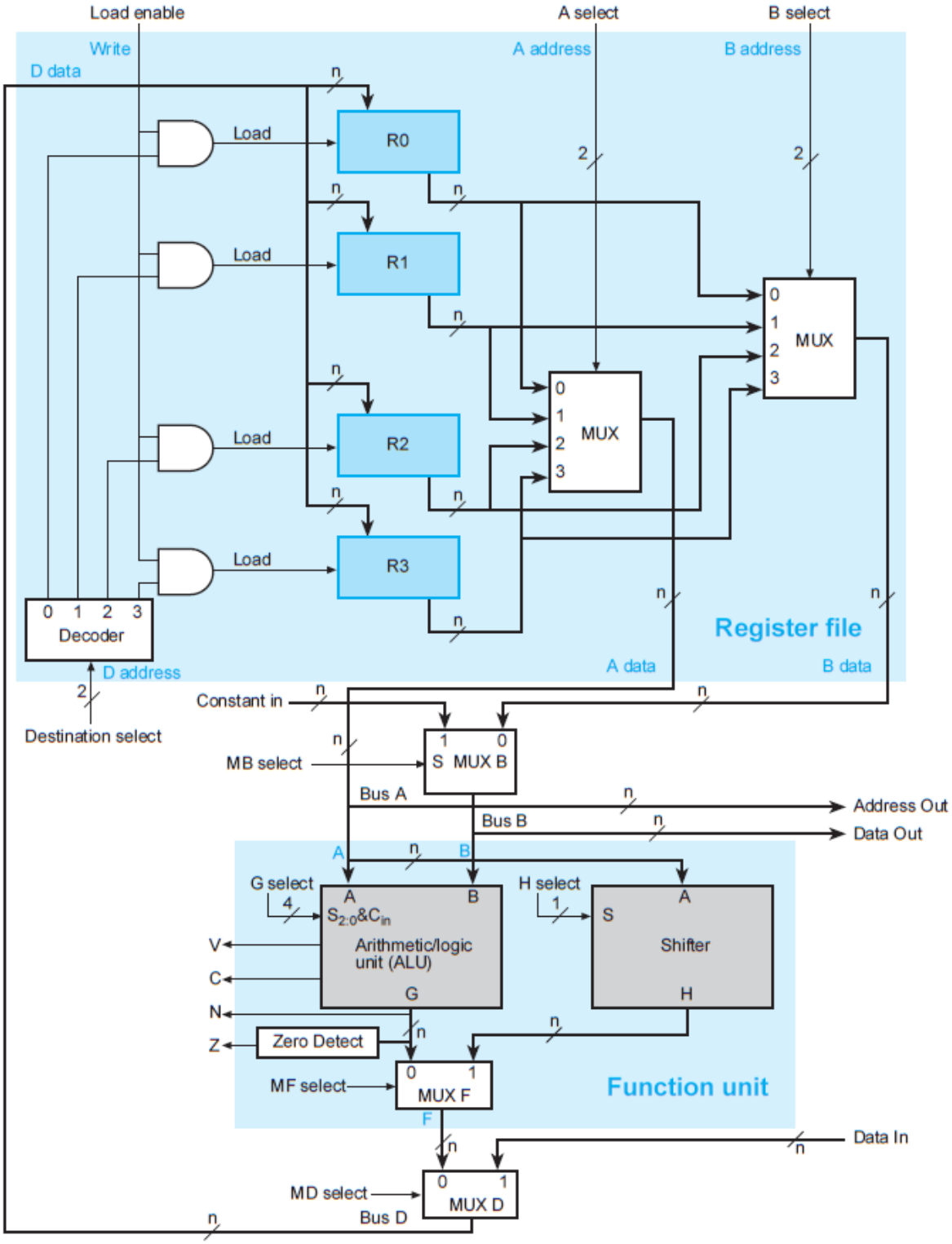
4 bitlik register'larla işlem yapılsa da gerçek bilgisayarlarda 32 bit veya daha fazlası kullanılmaktadır. Tabi böylesine büyük yapılar datapath tasarlama da farklı tekniklerin kullanılmasını gerektirecektir. İlk datapath resmindeki register file ve fuction unit'i bir blok halinde gösterirsek,



- V çıkış sinyali bize işlam sonrasında taşma durumu (overflow) varsa 1 değerini alarak taşma durumu olduğunu işaret eder
- C çıkış sinyali Cout'u temsil eder. Yani işlem sonrasında elde kalan değer olarak tanımlanır.
- N çıkış sinyali Function unit'ten çıkan işlem sonucunun negatif olması durumunda 1 olan pozitif olması durumunda 0 değerini veren bir sinyaldir

- Z çıkış sinyali işlem sonucunun 0 (zero) olduğu durumda 1 değerini veren bir çıkış sinyalidir.

Şimdi ilk datapath devresini yeniden irdeleyelim



Yukarıda anlattıklarımızı da göz önüne alarak şöyle bir doğruluk tablosu oluşturabiliriz

FS(3:0)	MF Select	G Select(3:0)	H Select(3:0)	Microoperation
0000	0	0000	XX	$F \leftarrow A$
0001	0	0001	XX	$F \leftarrow A + 1$
0010	0	0010	XX	$F \leftarrow A + B$
0011	0	0011	XX	$F \leftarrow A + B + 1$
0100	0	0100	XX	$F \leftarrow A + \overline{B}$
0101	0	0101	XX	$F \leftarrow A + \overline{B} + 1$
0110	0	0110	XX	$F \leftarrow A - 1$
0111	0	0111	XX	$F \leftarrow A$
1000	0	1X00	XX	$F \leftarrow A \wedge B$
1001	0	1X01	XX	$F \leftarrow A \vee B$
1010	0	1X10	XX	$F \leftarrow A \oplus B$
1011	0	1X11	XX	$F \leftarrow \overline{A}$
1100	1	XXXX	00	$F \leftarrow B$
1101	1	XXXX	01	$F \leftarrow \text{sr } B$
1110	1	XXXX	10	$F \leftarrow \text{sl } B$

Buradaki FS select, MF select, G select, ve H select den oluşmaktadır. FS select için çıkarımlarımız şöyle;

- FS select in en soldaki 2 biti 1 olduğunda MF select=1 olduğu gözüküyor.
- Eğer MF select=0 ise Fs selecti oluşturan kodlar aynen G select deki kodlardır.
- MF select=1 olduğunda FS selectin en soldaki 2 bitini 1 olduğunu belirtmiştik diğer 2 biti ise H select tarafından kullanılır

Burada gerekli boolean işlemleri yapıldığında

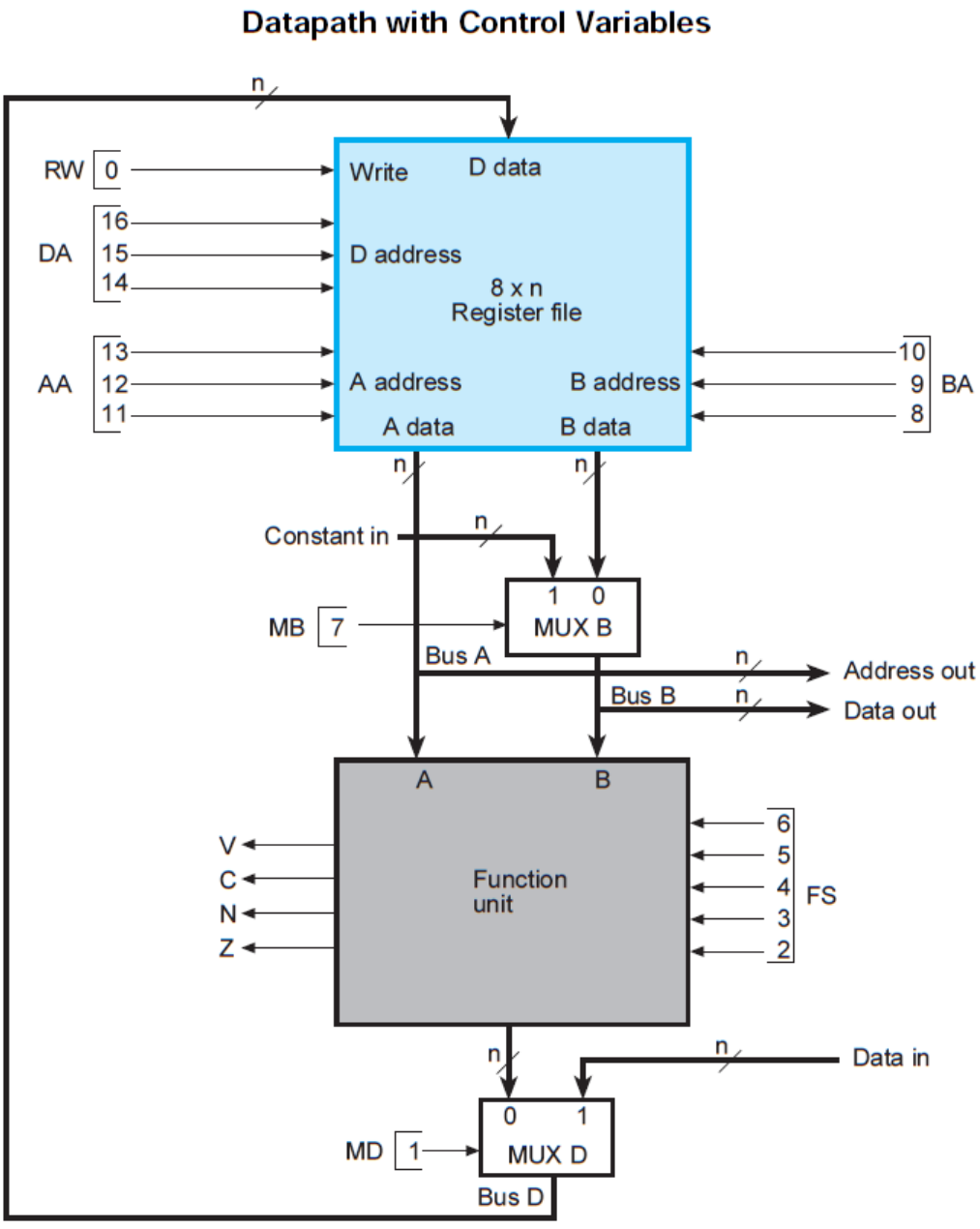
- MF=F3.F2
- G3=F3
- G2=F2
- G1=F1
- G0=F0
- H1=F1
- H0=F0

olduğunu test edebilirsiniz.

Not: Sadece 4 bitlik FS select ile MF, G ve H selectleri kontrol ederek function unit bloğunun ne yapacağına artık karar verebiliriz.

Control Word:

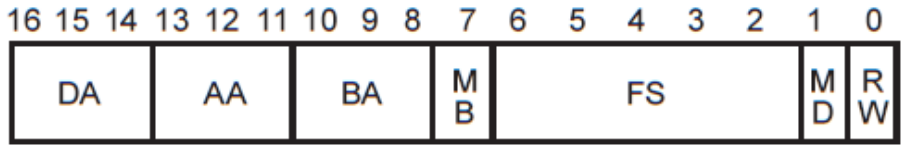
Şimdi datapath'i yeniden ele alalım,



(a) Block Diagram

Yukarıdaki datapath yapısında bulunan register file içerisinde R0'dan R7'ye olmak üzere 8 adet register bulunmaktadır. Bu register'lara 3'e 8 decoderle ulaşılabilir (ilk datapath resminde bu decoderler görülmektedir).

Toplama bakılacak olunursa resimdeki numaralardan belli olacağı üzere 16 adet binary giriş var bunlar yerlerine göre numaralandırılmıştır. Bu girişlerin hepsini bir araya getirerek *Control Word* oluşturulur. *Control Word* aşağıdaki biçimde gösterilir.



(b) Control word

Control word'deki bitlerin numaralarıyla datapath resminde ifade ettikleri kısımların aynı yer olduğunu fakedeceksiniz.

- DA , datanın yazılacağı registerın adresini gireceğimiz decoder'ın kodu
- AA, Bus A'ya konulacak datanın hangi register'dan geleceğini seçecek Mux'un Select bitleri
- BA, Bus B'ya konulacak datanın hangi register'dan geleceğini seçecek Mux'un Select bitleri
- MB, Bus A'dan gelen data ile herhangi bir sabitin mi yoksa B'den gelen data ile mi işleme sokulacağı karar veren Mux'un select biti
- FS, daha önce tabloda boolen ifade ile bulmuş olduğumuz FS Select girişleri
- MD, function unit'ten mi yoksa dışarıdan gelen bir bilginin mi hedefdeki register'a yazılacağını belirleyen Mux'un Select girişi
- RW, ise register dosyalarına yazma veya okuma işlemlerinin hangisinin yapılacağına karar verir.

Şimdi bu control word yardımıyla hangi işlemleri hangi kodlarla yazılacağına bakalım.

Encoding of Control Word for the Datapath

DA, AA, BA		MB		FS		MD		RW	
Function	Code	Function	Code	Function	Code	Function	Code	Function	Code
$R0$	000	Register	0	$F = A$	00000	Function	0	No write	0
$R1$	001	Constant	1	$F = A + 1$	00001	Data In	1	Write	1
$R2$	010			$F = A + B$	00010				
$R3$	011			$F = A + B + 1$	00011				
$R4$	100			$F = A + \overline{B}$	00100				
$R5$	101			$F = A + \overline{B} + 1$	00101				
$R6$	110			$F = A - 1$	00110				
$R7$	111			$F = A$	00111				
				$F = A \wedge B$	01000				
				$F = A \vee B$	01010				
				$F = A \oplus B$	01100				
				$F = \overline{A}$	01110				
				$F = \text{sr } A$	10000				
				$F = \text{sl } A$	10001				

Control Word Example

100 100 001 0 1001 0 1

R1 **00100000**
R4 **01010100**

$R4 \leftarrow R4 \vee R1 \leftarrow 01110100$ 't'

101 101 001 0 1010 0 1

R1 **00100000**
R5 **01001100**

$R5 \leftarrow R5 \oplus R1 \leftarrow 01101100$ 'l'

001 001 000 0 1011 0 1

$R1 \leftarrow \overline{R1} \leftarrow 1101111$

001 001 000 0 0001 0 1

$R1 \leftarrow R1 + 1 \leftarrow 11100000$

110 110 001 0 0101 0 1

R6 **01000001**

$R6 \leftarrow R6 + \overline{R1} + 1 \leftarrow 0110000$ 'a'

111 111 001 0 0101 0 1

R7 **01001001**

$R7 \leftarrow R7 + \overline{R1} + 1 \leftarrow 0110100$ 'i'

011 111 000 0 0000 0 1

$R1 \leftarrow R7 \leftarrow 01101001$ 'i'

Örnek:

R1 ← R2 + R3' + 1 işlemini yapabilmek için gereken *Control Word*'e yazılacak kodları bulun.

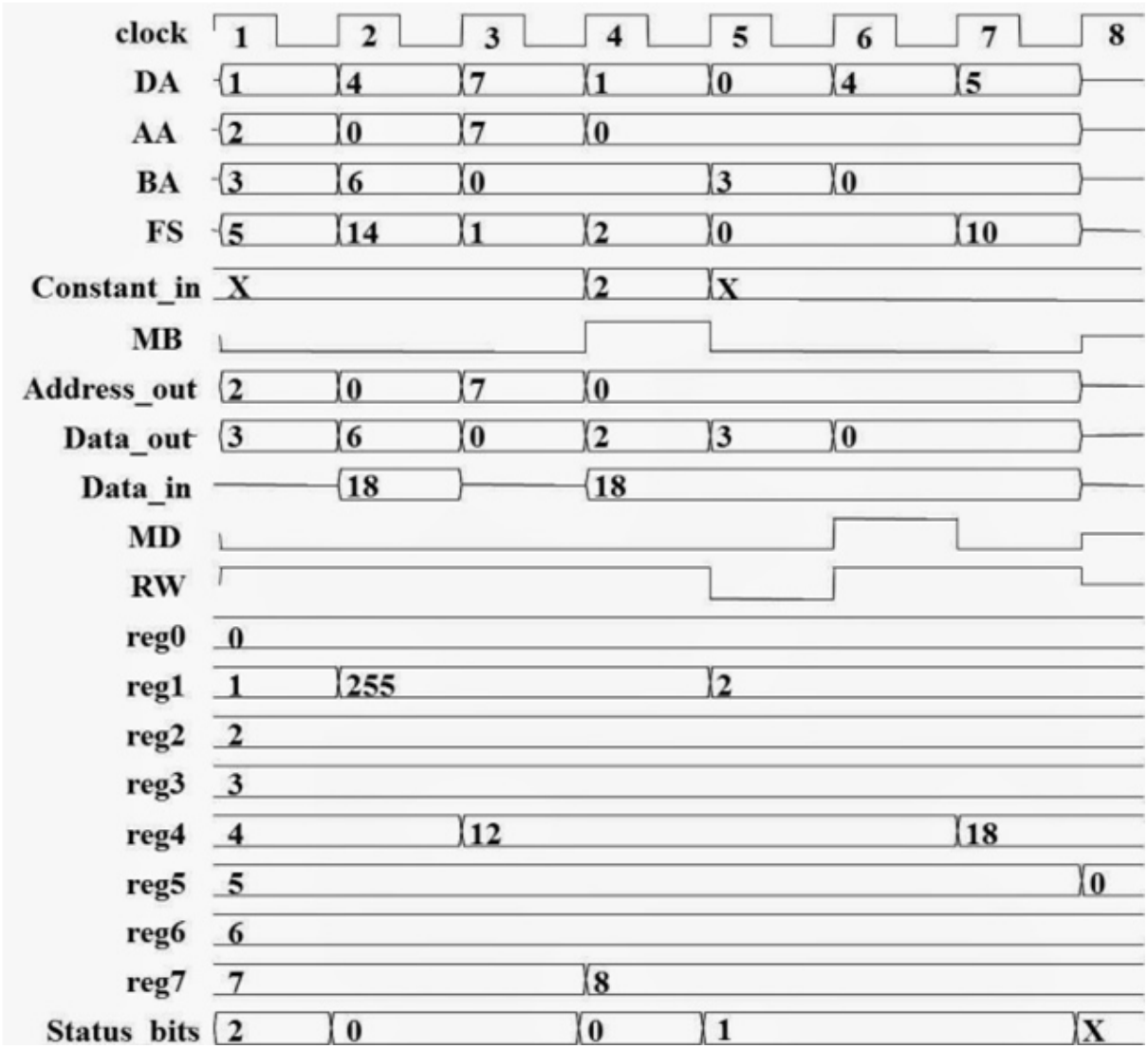
- Yazılacak adres R1 olduğu için Destination Register(hedef Register) R1 olacaktır. Tablodan R1'in koduna bakıldığında, DA=001 olacağı belirlenir.
- Bus A dan gelecek data R2 dir. Yine R2'nin tablodaki koduna bakıldığında, AA=010 olacaktır.
- Bus B den gelecek data da R3 dür ve R3'ün tablodaki koduna bakıldığında, BA=011 elde edilir.
- Daha sonra dışarıdan herhangi bir sabitle işlem olmadığı için MB=0.
- Yapılacak işlem R1 ← R2 + R3' + 1 dir. Function Code'dan bu işlemin hangi kodda sağlandığına bakılır, FS=0101
- Dışarıdan herhangi bir data girişi olmadığı için, MD=0
- Yazma işlemi yapacağımız için, RW=1 olacaktır.

Önce işlemler kod olarak değil de isim olarak verilir ve bu tablonun altına da o isimlerin gerektirdiği kodları benzer bir tabloda verilimiş olur.

Micro-Operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 - R3$	$R1$	$R2$	$R3$	Register	$F = A + \overline{B} + 1$	Function	Write
$R4 \leftarrow$ sl R6	$R4$	—	$R6$	Register	$F =$ sl B	Function	Write
$R7 \leftarrow R7 + 1$	$R7$	$R7$	—	Register	$F = A + 1$	Function	Write
$R1 \leftarrow R0 + 2$	$R1$	$R0$	—	Constant	$F = A + B$	Function	Write
Data out $\leftarrow R3$	——		$R3$	Register	—	—	No Write
$R4 \leftarrow$ Data in	$R4$	——		—	—	Data in	Write
$R5 \leftarrow 0$	$R5$	$R0$	$R0$	Register	$F = A \oplus B$	Function	Write

Micro-Operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 - R3$	001	010	011	0	0101	0	1
$R4 \leftarrow$ sl R6	100	XXX	110	0	1110	0	1
$R7 \leftarrow R7 + 1$	111	111	XXX	0	0001	0	1
$R1 \leftarrow R0 + 2$	001	000	XXX	1	0010	0	1
Data out $\leftarrow R3$	XXX	XXX	011	0	XXXX	X	0
$R4 \leftarrow$ Data in	100	XXX	XXX	X	XXXX	1	1
$R5 \leftarrow 0$	101	000	000	0	1010	0	1

Datapath'in çalışmasının zaman diagramı üzerinde gösterimi:

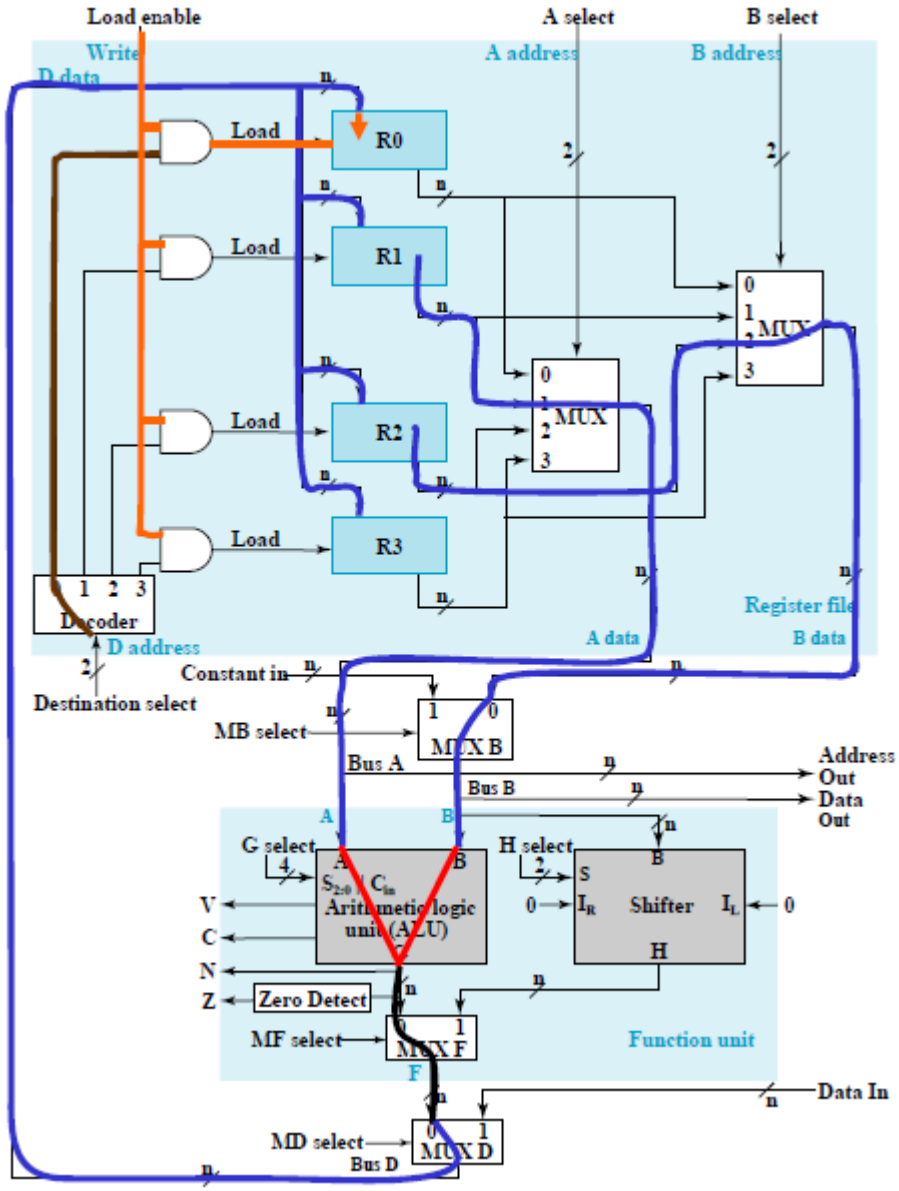


Not: Bu gösterimde değerler binary olarak değilde desimal yazılmıştır

Datapath Örneği:

- Dört paralel register yüklemesi
- İki mux tabanlı register seçici
- Register Hedef kod çözücü
- Harici sabit giriş için Mux B
- Harici adres ve veri çıkışlarına sahip A ve B busları
- Çıkış seçimi için Mux F'li ALU ve Shifter
- Harici veri girişi için Mux D
- V, C, N, Z durum bitlerini oluşturmak için mantık

Mikro İşlem Gerçekleştirme:
R0 ← R1 + R2



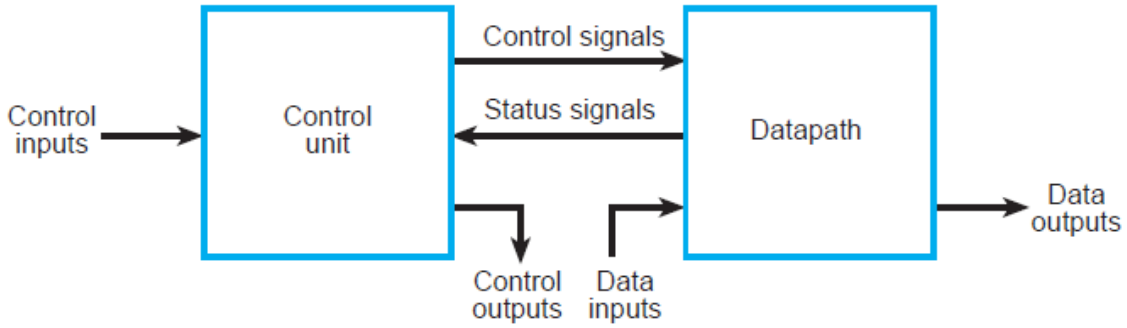
- R1 içeriğini Bus A'ya yerleştirmek için A'ya 01 uygulanır.
- R2'nin içeriğini B verilerine yerleştirmek için B'ye 10 uygulanır ve B verilerini B veriyoluna yerleştirmek için MB'ye 0 uygulanır.
- Eklemeyi gerçekleştirmek için 0010'dan G'ye seçilir, G = Bus A + Bus B
- G değerini BUS D üzerine yerleştirmek için MF seçimine 0 ve MD seçimine 0 uygulanır.
- R0'a Yük girişini etkinleştirmek için Hedefe 00 uygulanır.
- Yük girişini R0'a 1'e zorlamak için Yük Etkinleştir'e 1 uygulanır, böylece R0, saat darbesine yüklenir (gösterilmemiştir)
- Genel mikro işlem 1 saat döngüsü gerektirir.

Control Unit:

Genel olarak bir senkron digital sistemde tüm register'ların zamanlaması bir temel clock generatöründen sağlanır. Bu sistemde clock darbeleri tüm flip-flop'lara ve register'lara uygulanır. Register'ların veya flip-flop'ların durumunu her clock darbesi sırasında değişime uğratmayı engellemek için ise register ve flip-flop'lara bir Enable, Disable ve Load girişleri eklenir. Eğer Enable aktifse Register'dan data okunabilir, Eğer Load aktifse register'a data yüklemesi yapılabilir.

Sistem tasarımlarında, digital sistemlerdeki control unit yapısı iki ayrı türden oluşur, Bunlardan birincisi programlanabilen sistem diğeri ise programlanamayan sistem. Programlanabilir sistemlerde giriş kısımları sıralı bilgilerden oluşur. Programlanamayan sistemler de ise register'da toplanan bilgiyle alakası olmadan tasarlanmış bir devre halinde gerekli işlemler yapılır.

Interaction between Datapath and Control Unit



5. Kaynaklar

1) *http://history.acusd.edu/gen/recording/computer1.html*
2) *http://www.cs.virginia.edu/brochure/museum.html*
3) *http://www.columbia.edu/acis/history/650.html*
4) *http://www.piercefuller.com/collect/pdp8.html*
5) *http://www.computer50.org/kgill/transistor/trans.html*
6) *The History of The Microprocessor, Bell Labs Technical Journal, Autumn, 1997*
7) *http://www.intel.com*
8) *www.cs.sjsu.edu/faculty/lee/chapter3_presentation2.ppt*
9) *https://profs.basu.ac.ir/.../722.1869.file_ref.1998.2468.ppt*
10) *www.abandah.com/.../22446_S11_Intro_to_microprocessor...*
11) *rise.cse.iitm.ac.in/people/faculty/kama/prof/x86_1.ppt*
12) *www.cse.unsw.edu.au/~cs2121/.../week3_notes.pp*
13) *https://users.cs.jmu.edu/.../IntelProcessors4004ToPentiumPr...*
14) *Microprocessor, Atul P. Godse, Deepali A. Gode, Technical publications, Chap 11*
15) *The 8088 and 8086 Microprocessors: Programming, Interfacing, Software, Hardware, and Applications (4th Edition) Paperback – Aug 29 2002 by Walter A. Triebel (Author), Avtar Singh (Author).*
16) *Dandamudi S., 2003. Fundamentals of Computer Organization and Design, Springer-Verlag, Heidelberg*
17) *Mano, M.M., 2007. Bilgisayar Sistemleri Mimarisi, Literatür Yayıncılık, İstanbul*
18) *Buzluca F., "Bilgisayar Mimarisi Ders Notları", Bilgisayar Mühendisliği Bölümü, İTÜ Bilgisayar ve Bilişim Fakültesi*
19) ***https://tr.wikipedia.org/wiki/Aritmetik_mant%C4%B1k_birimi***
20) *CSE 675.02: Introduction to Computer Architecture, Slides by Gojko Babiç*