
# BLM4021 Gömülü Sistemler

Doç. Dr. Ali Can KARACA

*ackaraca@yildiz.edu.tr*

Yıldız Teknik Üniversitesi – Bilgisayar Mühendisliği

# Sunum 3 – Mikrodenetleyiciler (PIC ve MSP430)

- Laboratuvar Hatırlatma
- Genel Mikrodenetleyici Yapısı
- 16/32/64 bitlik Mikrodenetleyiciler
- PIC ve özellikleri
- MSP430 ve özellikleri

# Ders Materyalleri

***Gerekli Kaynaklar:***

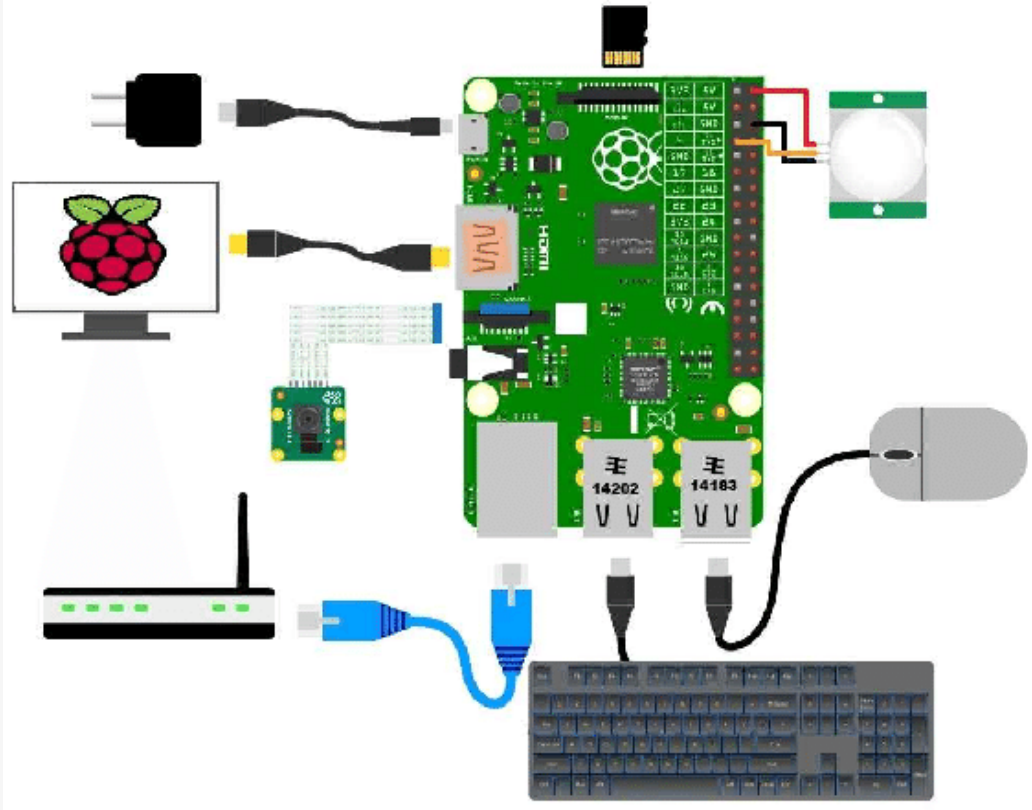
- Derek Molloy, Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux, Wiley, 2016.
- M. Wolf, Computers as Components: Principles of Embedded Computing System Design, Elsevier, 2008.

***Yardımcı Kaynaklar:***

- P. Membrey, D. Hows, Learn Raspberry Pi 2 with Linux and Windows 10, Apress, 2015.
- F. Vahid, T. Givargis, Embedded System Design, 1999.
- P. Koopman, Better Embedded Systems Software, Drumadrochit Education Pub., 2010.
- O. Urhan, Gömülü Sistem Lisansüstü Ders Notları, 2018.
- P. Jones, Embedded Systems Design, CPRE488 Lecture Notes, IOWA State University.

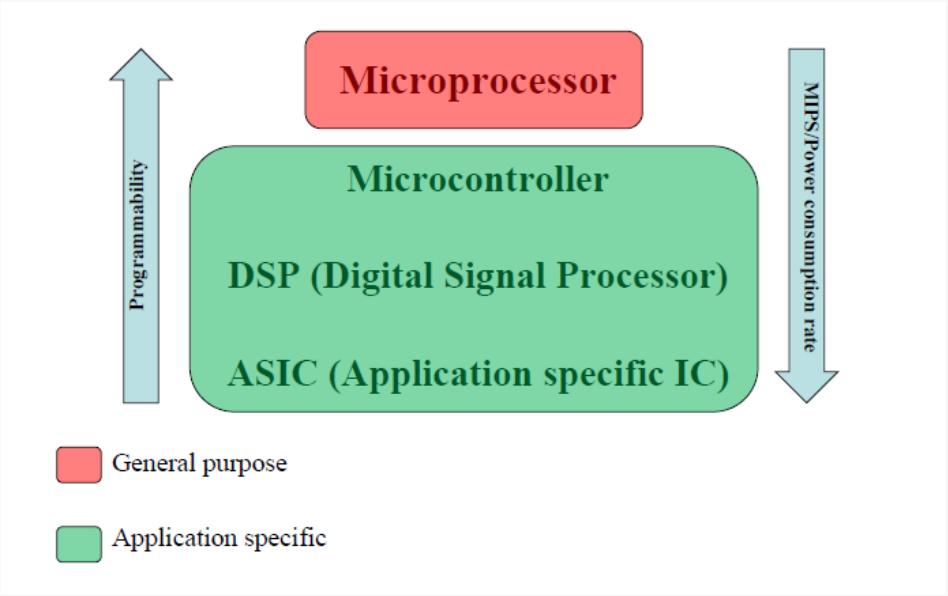
# Hatırlatma: Raspberry Pi ile çalışmak için gerekenler

1. Raspberry Pi Kiti (Laboratuvarımızda var.)
2. En az 2A'lik bir güç adaptörü (Laboratuvarımızda var.)
3. Mikro USB kablo
4. SD kart (8-32 GB)
5. HDMI ekran kablosu
6. Monitor, klavye ve Mouse
7. Ethernet kablosu



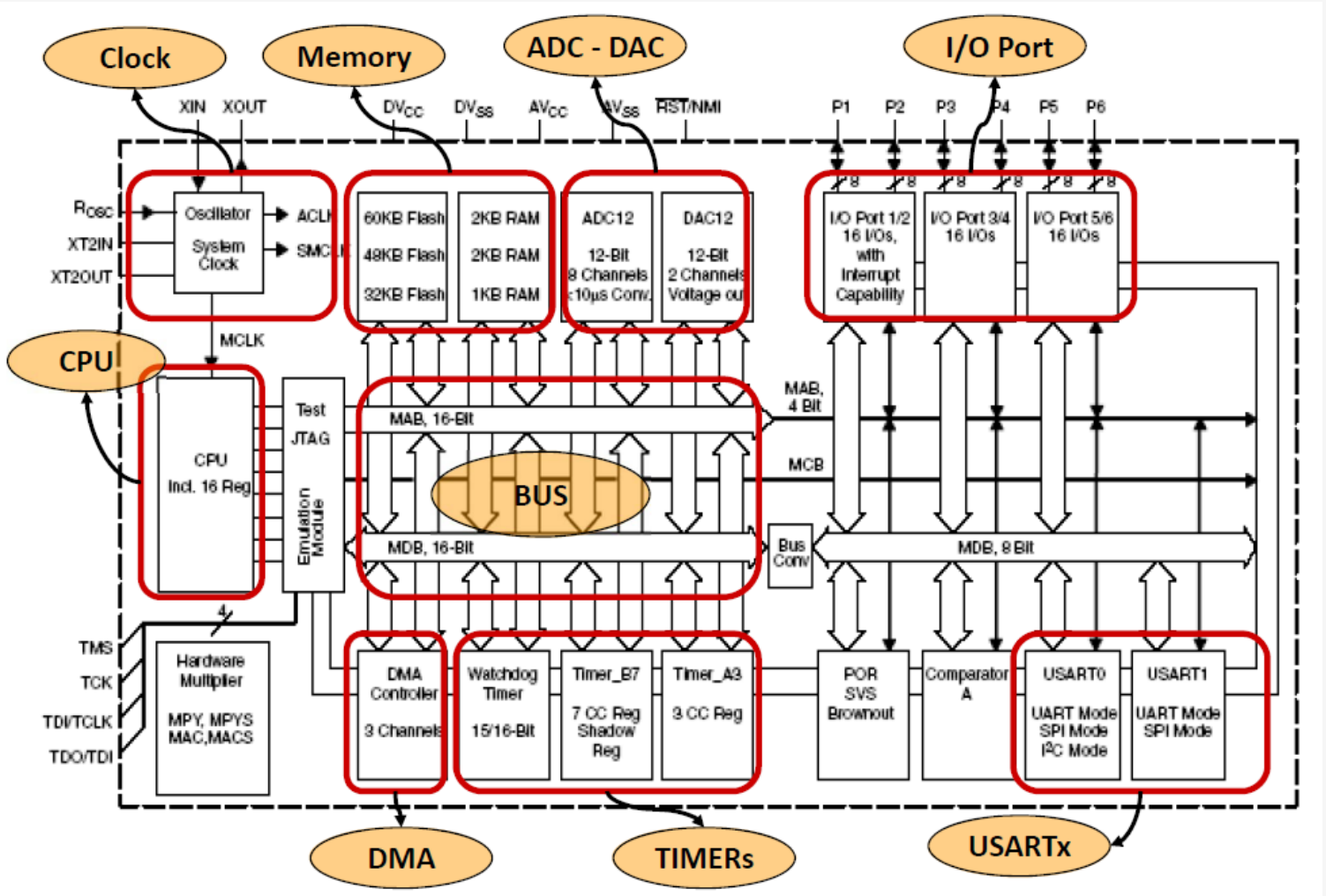
# Neden mikrodenetleyiciler?

- Fazla sayıda ve çeşitli giriş-birimleri (DAC, ADC, I2C, PWM vb.),
- Düşük güç tüketimi,
- Kolay programlanabilir,
- Devrelere basitçe entegre edilebilir,
- Tek amaçlı,
- Düşük ücretli,
- Küçük hacme sahip.




Credits D. Balsamo, M. Magno, L. Benini

# Örnek bir MCU-Temelli Sistem Mimarisi




- ADC (Analog-to-Digital Converter)
- DAC (Digital-to-Analog Converter)
- BUS (veri/adres yolu)
- Timer
- USART (Serial Comm.)
- DMA (Direct Memory Access)
- I/O Port (Input-Output Port)

Credits D. Balsamo, M. Magno, L. Benini

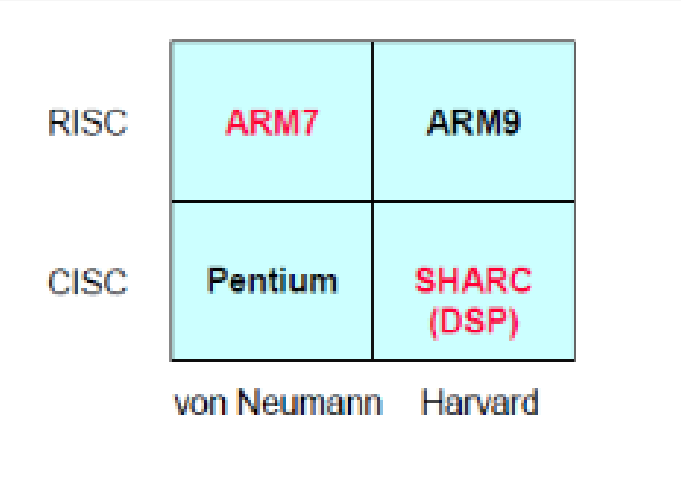
# Merkezi İşlem Birimi (CPU) Sınıflandırılması

Komut yapıları, sayıları ve adresleme tiplerine göre:

- CISC (Complex Instruction Set Computer)
- RISC (Reduces Instruction Set Computer)

Komut ve belleğe erişim ile ilgili:

- Von Neumann mimarisi
- Harvard mimarisi




Register/veriyoluna göre:

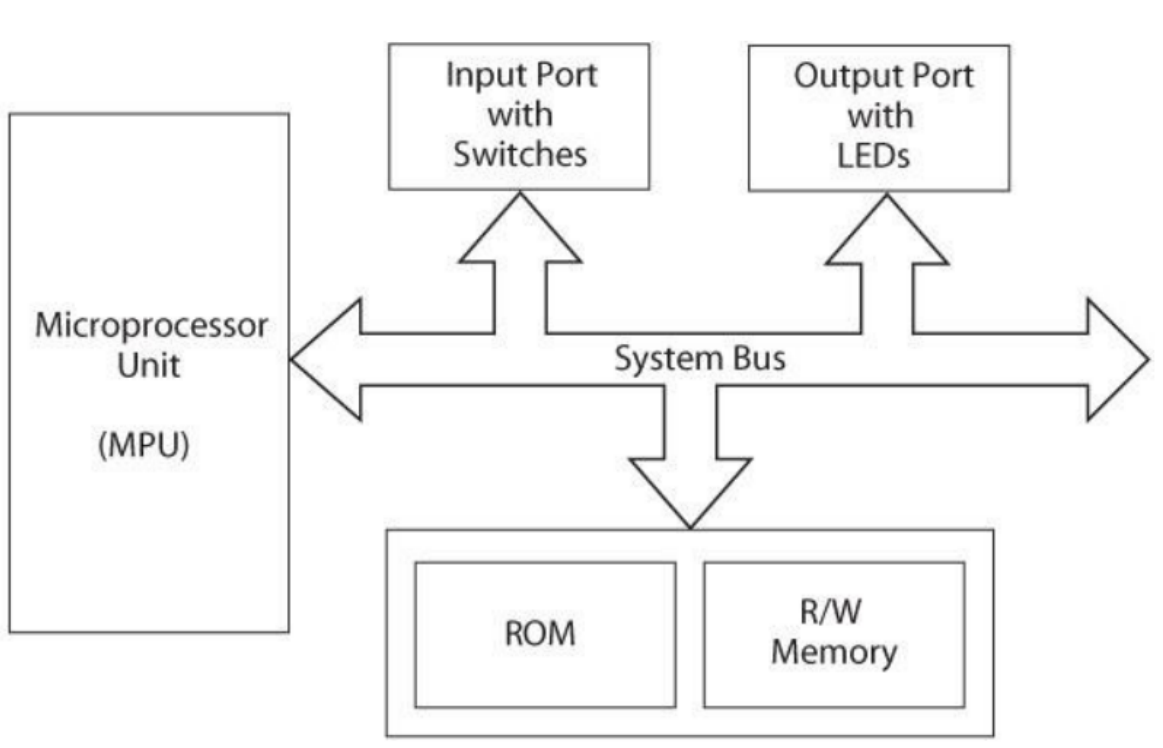
- 8 bit (örn. PIC), 16 bit (örn. MSP430), 32 bit (örn. ARMV7), 64 bit (örn. ARM v8)

# Mikroişlemci Temelli Sistemlerde Sistem Yolu

Bir mikroişlemci temelli sistemde amaç, mikroişlemci, bellek ve giriş/çıkış portlarındaki bilgileri kullanarak verileri işlemektir.

- Giriş portundaki bilgilerine ulaşabilmek,
- Çıkış portuna bilgi yazmak,
- Gerekli olan programı işletebilmek için bilgi alışverişi gerekli !!

<< Sistem Yolu >>



# Mikroişlemci Temelli Sistemlerde Adres, Veri ve Kontrol Yolu

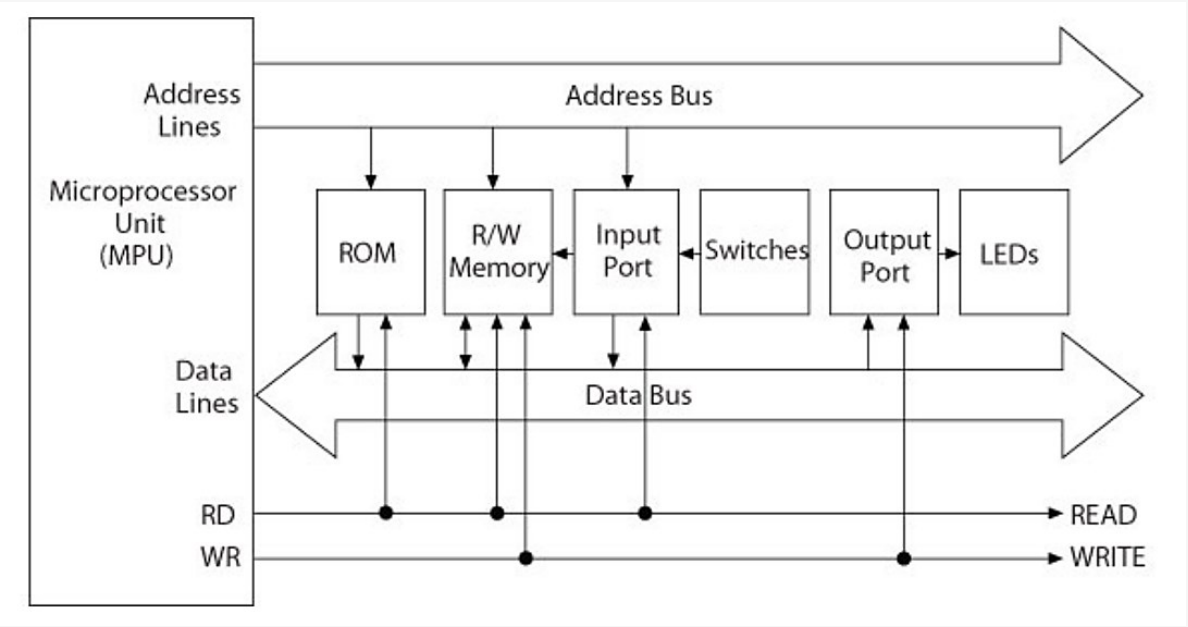
CPU ile diğer bileşenlerin haberleşmesi için kullanılan sistem yolu üçe ayrılır:

## 1. Adres Yolu (Adress Bus):

Tek yönlüdür. Hafıza ve I/O adreslerini içerebilir.

## 2. Veri yolu (Data Bus):

İki yönlüdür. İkili veri ve komutları transfer eder.




## 3. Kontrol yolu (Control Bus):

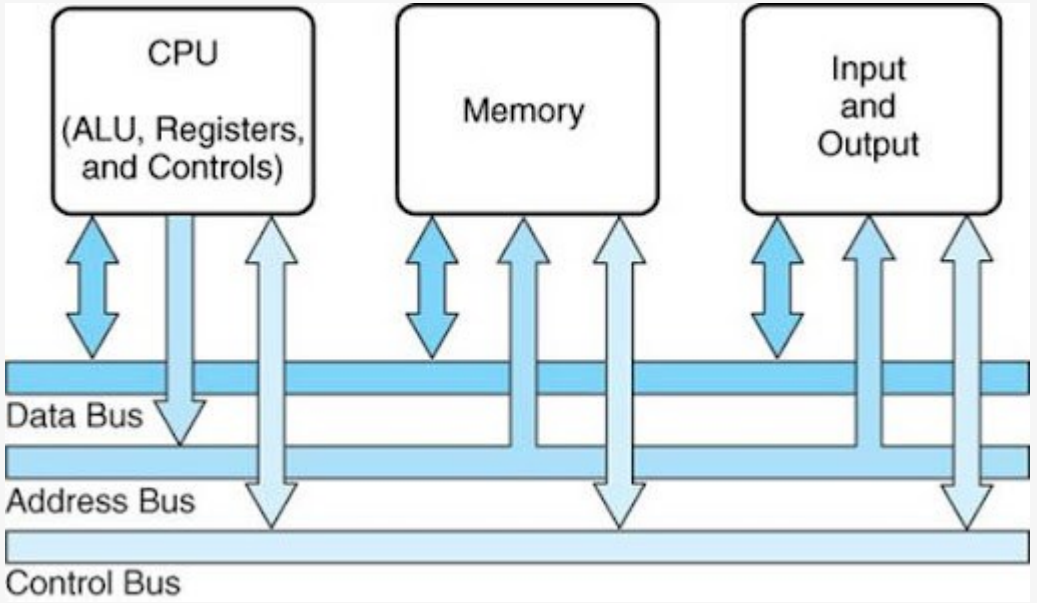
Belleğe yazma ve bellekten okuma ile isteklerini iletir.

# Mikroişlemci Temelli Sistemlerde Adres, Veri ve Kontrol Yolu

CPU ile diğer bileşenlerin haberleşmesi için kullanılan sistem yolu üçe ayrılır:

1. Adres Yolu (Adress Bus):

- Tek yönlüdür. Hafıza ve I/O adreslerini içerebilir.
- Verinin nereden geleceğini veya nereye yazılacağını gösterir.
- 16bitlik bir adres yolu varsa işlemci 64 KB bellek alanını adresleyebilirken;
- 32 bitlik bir adres yolu varsa işlemci 4 GB bellek alanını adresleyebilir.



# Mikroişlemci Temelli Sistemlerde Adres, Veri ve Kontrol Yolu

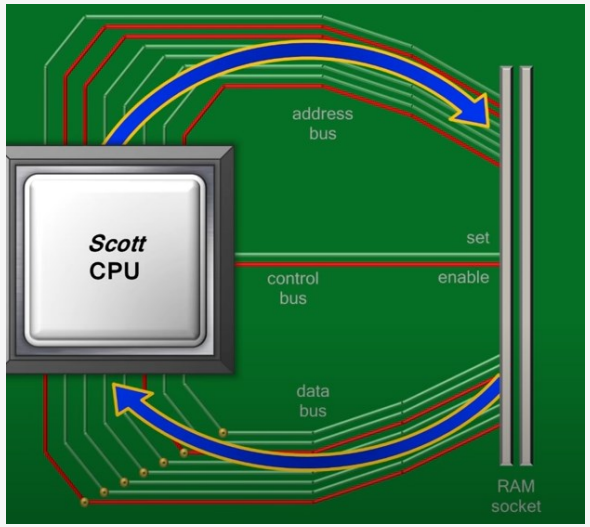
CPU ile diğer bileşenlerin haberleşmesi için kullanılan sistem yolu üçe ayrılır:

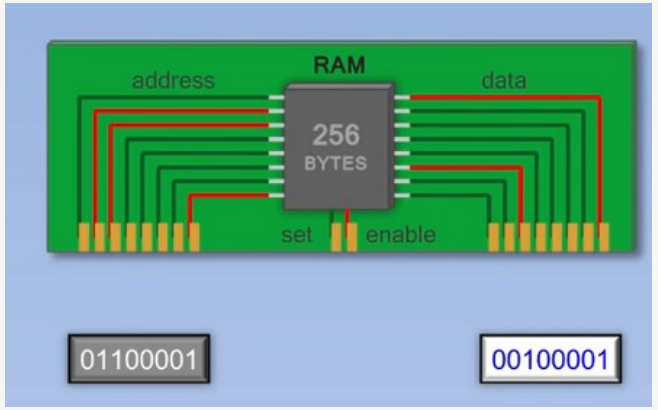
## 2. Veri Yolu (Data Bus):


- Çift Yönlüdür.
- Veri yolunun genişliği, bir çevrimde işlemcinin ne kadarlık bir veriyi okuyabileceğini/yazabileceğini gösterir.
- 8 bitlik mikroişlemci -> 8 bit veri yoluna sahiptir.
- Daha geniş veriyolu –> daha pahalı fakat daha hızlı bir CPU

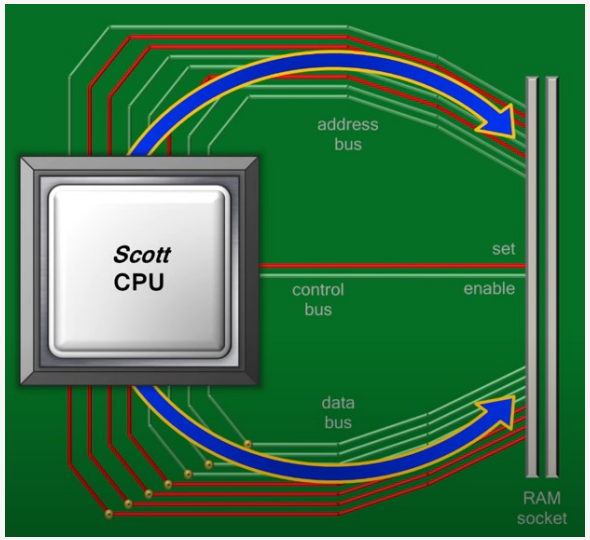
# Mikroişlemci Temelli Sistemlerde Adres, Veri ve Kontrol Yolu

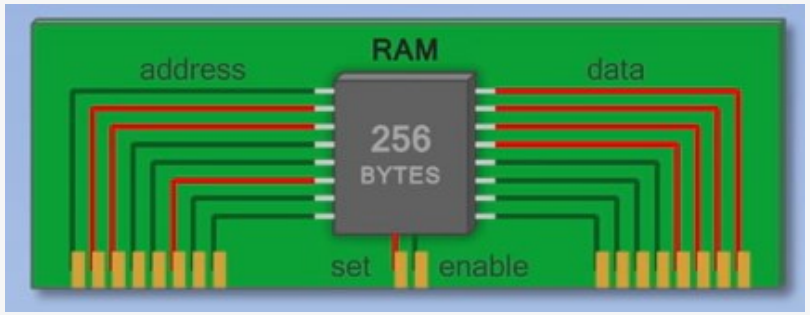
Veri Okuma:




Veri Kaydetme:

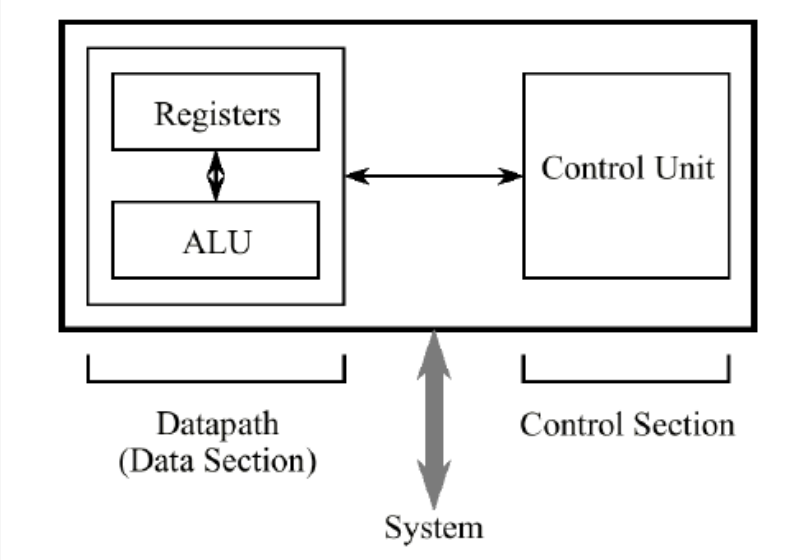


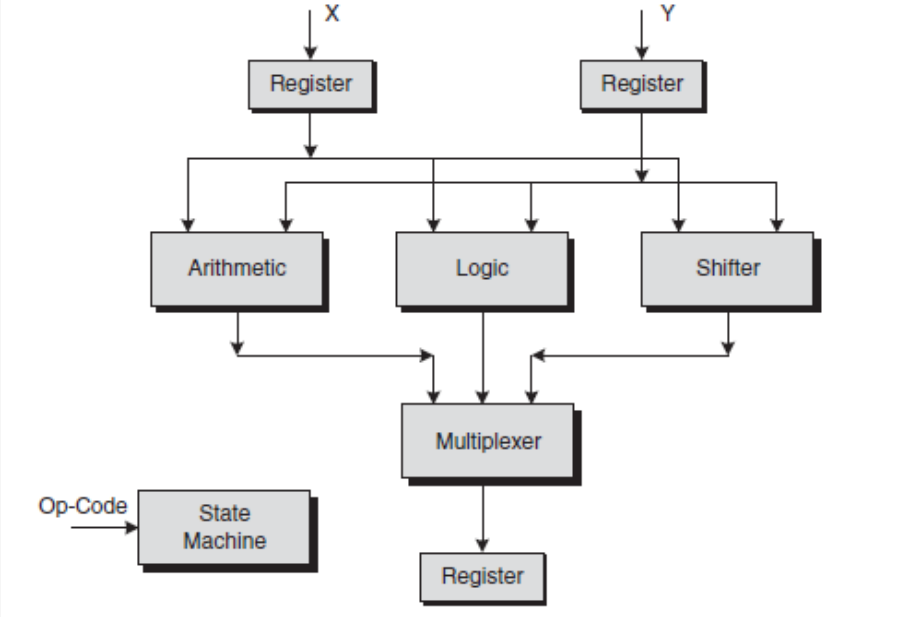



# Merkezi İşlem Birimi (CPU) içeriği

Merkezi işlem birimi: register, ALU ve kontrol biriminden oluşur.

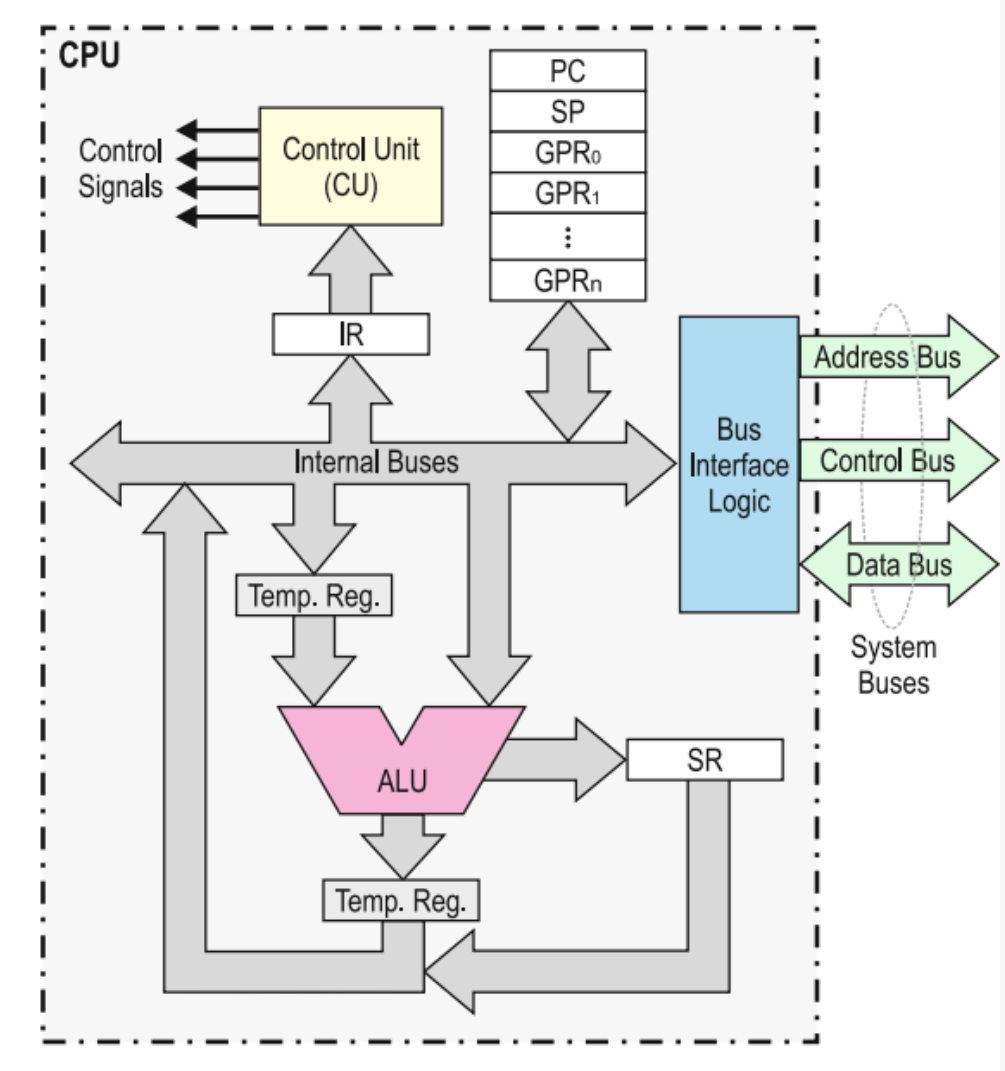
Komutları yorumlar ve register transferlerini yürütür.




ALU: Aritmetik Lojik Birim

# CPU: Ayrıntılı yapı




Datapath: ALU, iç veri yolu, registerlar, donanımsal çarpıcılar ve diğer bileşenleri içerir.

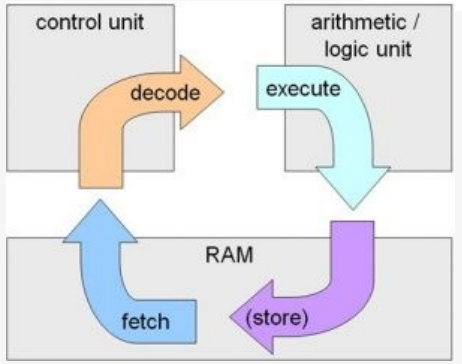
Control Unit: İşlemcinin kalbi olup komutları çalıştırmayı ve bu süreçteki tüm organizasyonu sağlar.

Bir komut en temelde dört aşamada işletilir. Buna pipeline adı verilir.

- Al getir (Fetch instruction),
- Komutu çöz (Decode the instruction),
- Komutu çalıştır (Execute the instruction),
- Geri yaz (Write back)

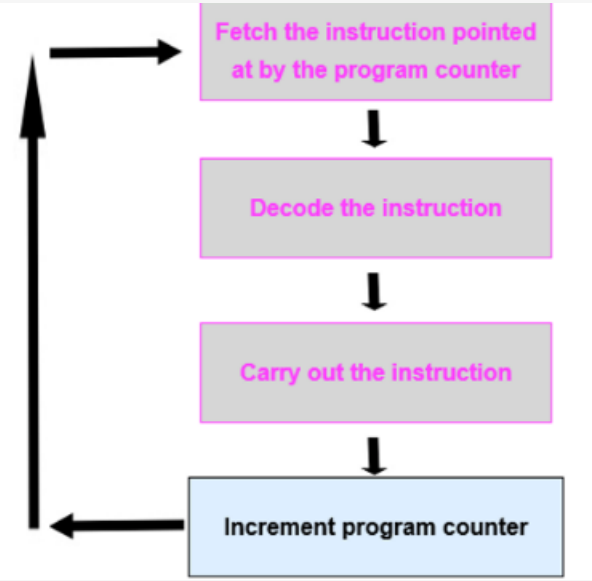
Credits M. Jimanez, R. Palomera, I. Couvertier

# Komut İşletimi (4-aşamalı pipeline)




Bir komut en temelde dört aşamada işletilir. Buna pipeline adı verilir.

- Al getir (Fetch instruction),
- Komutu çöz (Decode the instruction),
- Komutu çalıştır (Execute the instruction),
- Geri yaz (Write back)



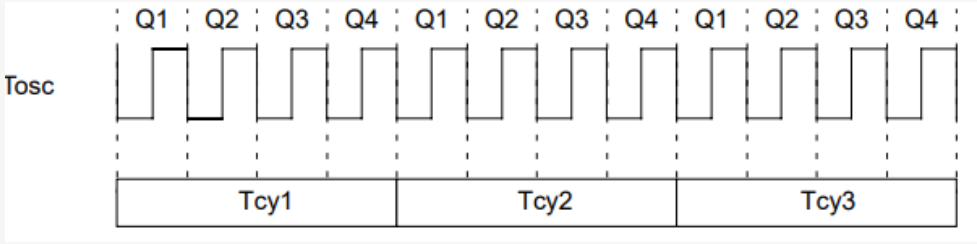

Program sayacı her başa sardığında artar, böylelikle komutların arka arkaya işletilmesi sağlanır.

# Komut İşletimi (4-aşamalı pipeline)

Bir komut en temelde dört aşamada işletilir. Buna pipeline adı verilir.

- Al getir (Fetch instruction),
- Komutu çöz (Decode the instruction),
- Komutu çalıştır (Execute the instruction),
- Geri yaz (Write back)

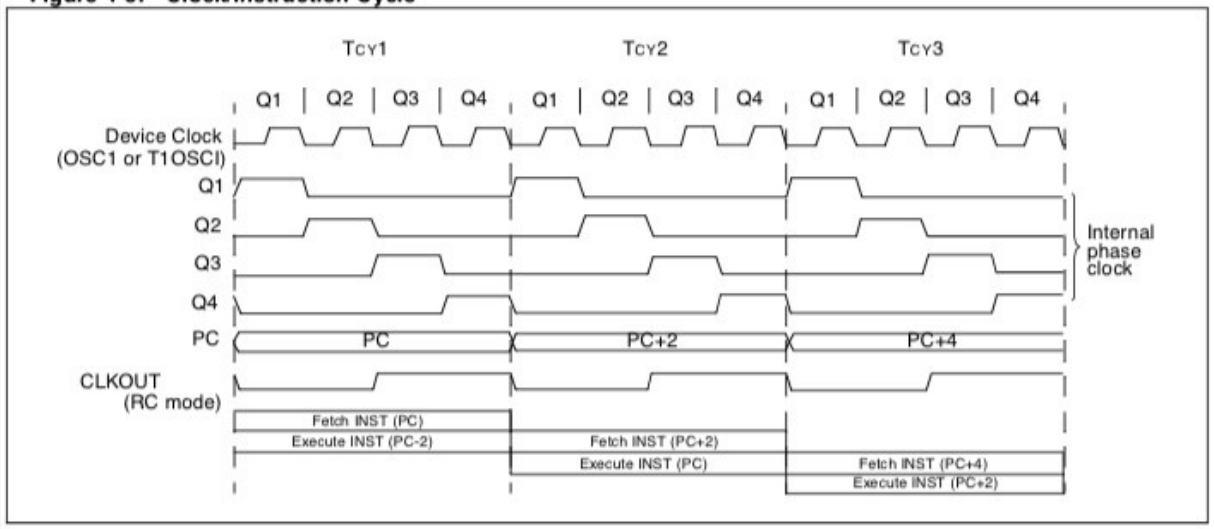
Dolayısıyla, tek bir komutu işletmek için 4 clock gerekecektir.



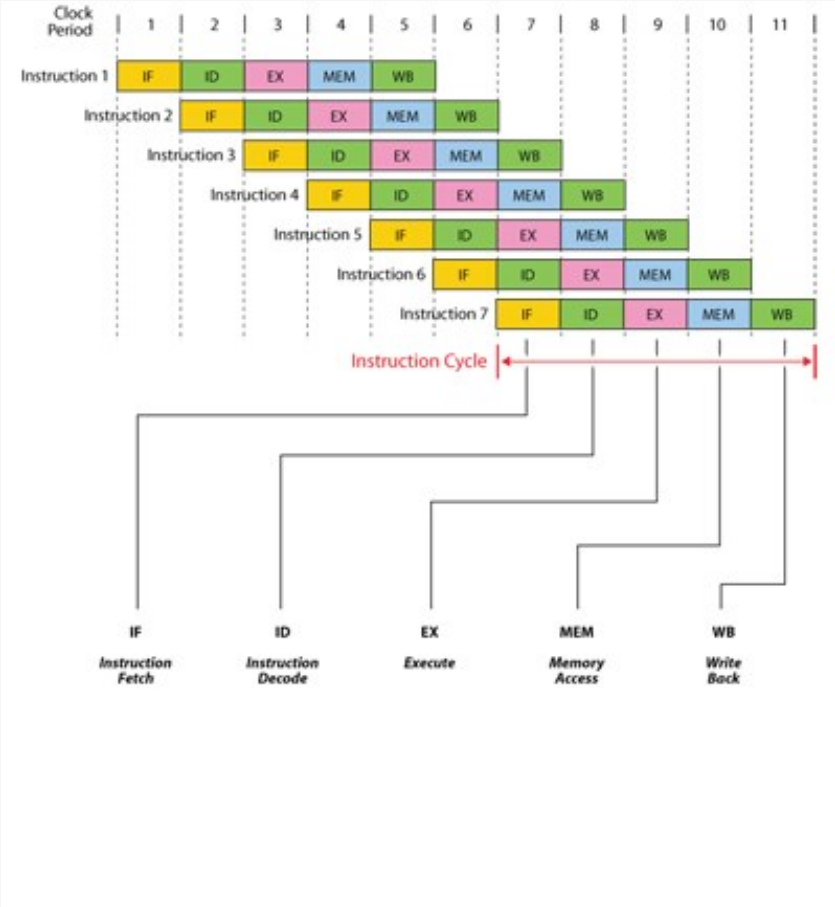

**4.2 Clocking Scheme/Instruction Cycle**

The clock input is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3 and Q4. Internally, the program counter is incremented every Q1, and the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are illustrated in Figure 4-3 and Example 4-1.

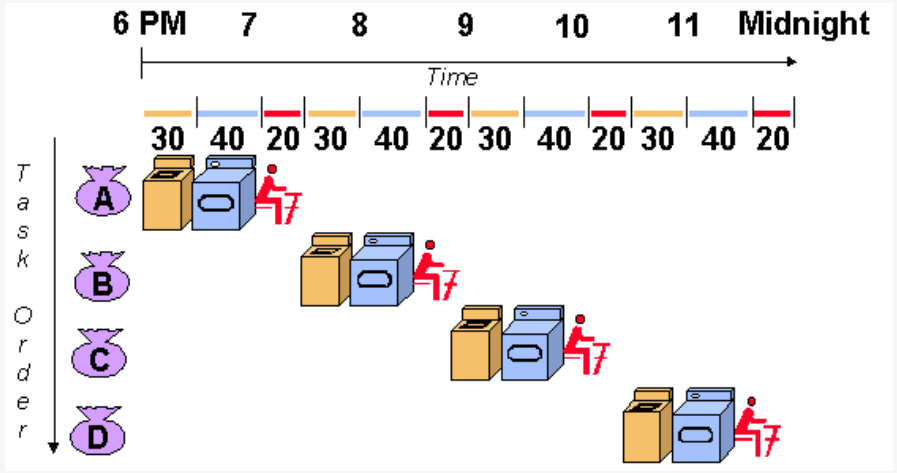
**Figure 4-3: Clock/Instruction Cycle**

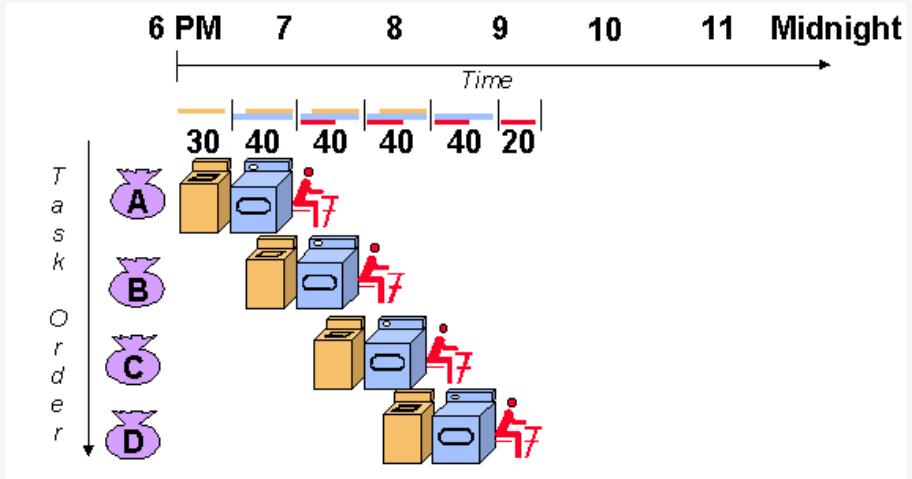


# 5-stage Pipeline




*Pipeline bize ne kazandırıyor?*



# Örnek: Komut Seti-PIC (Instruction Set from PIC)

- Format: **op k**
  - **op**: operation
  - **k**: literal, an 8-bit if data or 11-bit if address

- Examples:

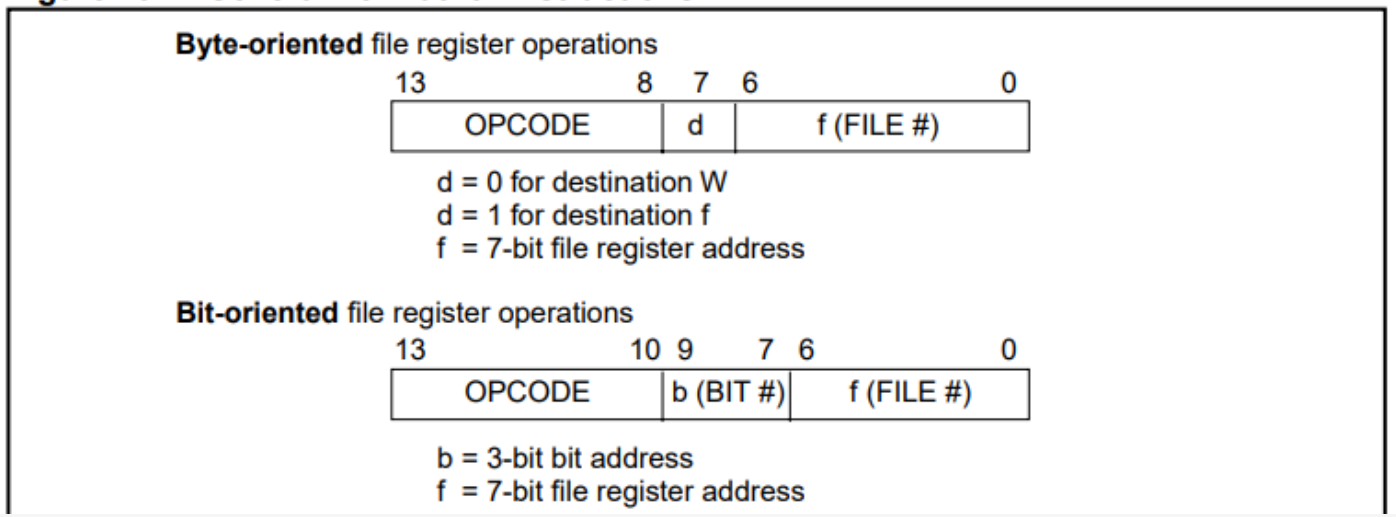
```
addlw   5
```

Adds to the working register the value 5.

```
call    9
```

Calls the subroutine at address 9.

**Figure 29-1: General Format for Instructions**




-> İşlem kodu + adres

# Animasyon (Tom Scott's Youtube Page)

# Bir işlemciyi 8-bit, 16-bit veya 32 bit olduğunu nasıl anlarız?

- The size of the registers?
- The size of the address bus?
- The size of the data bus?
- The size of the ALU (integer math unit)?
- The size of the PC (program counter)?

***Answer:***

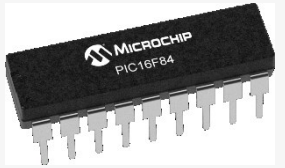
- On modern systems it typically is the integer registers, as well as the maximum size of a memory pointer (which typically is the same as the integer register size)
- On many systems though it is not as clear cut.

- A "pure" 8-bit system would have 8-bit registers (0-255), 8-bit ALU, and an 8-bit data bus.
- However an 8-bit address bus (only 256 bytes of RAM) is too limiting so most 8-bit processors (6502, z80, 8080, etc) had 16-bit address busses, 16-bit PCs, and often 16-bit register capability

Credit by Vince Weaver

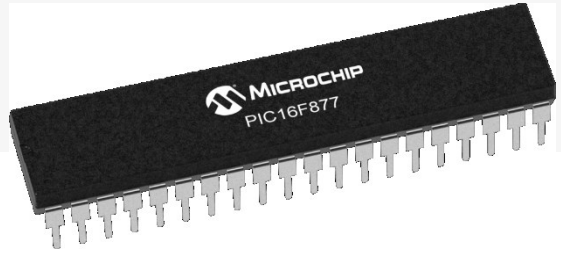
# PIC mikrodenetleyicileri (8-bit)

- Peripheral Interface Controlled (PIC), MicroChip firmasına aittir.




Genel Özellikleri:

- Düşük ücretli ve Harvard mimarisindedir.
- 50den az komut setine sahiptir (RISC).
- 8 bit veri yoluna sahiptirler.
- Tek accumulator registera sahip bir işlemci ailesidir.
- Az sayıda adresleme modu içerirler (RISC).
- Farklı pin sayılarında üretilebilirler.

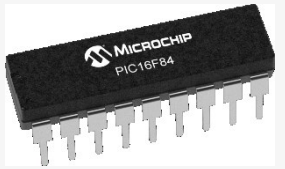


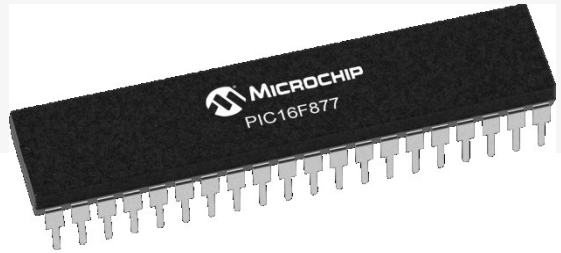

# PIC mikrodenetleyicileri

- Peripheral Interface Controlled (PIC), MicroChip firmasına aittir.

**The PIC Family: Speed**

- PICs require a clock to work.
  - Can use crystals, clock oscillators, or even an RC circuit.
  - Some PICs have a built in 4MHz RC clock
    - Not very accurate, but requires no external components!
  - Instruction speed = 1/4 clock speed
  - All PICs can be run from DC to their maximum specified speed:
    - 12C50x 4MHz
    - 12C67x 10MHz
    - 16Cxxx 20MHz
    - 17C4x / 17C7xxx 33MHz
    - 18Cxxx 40MHz

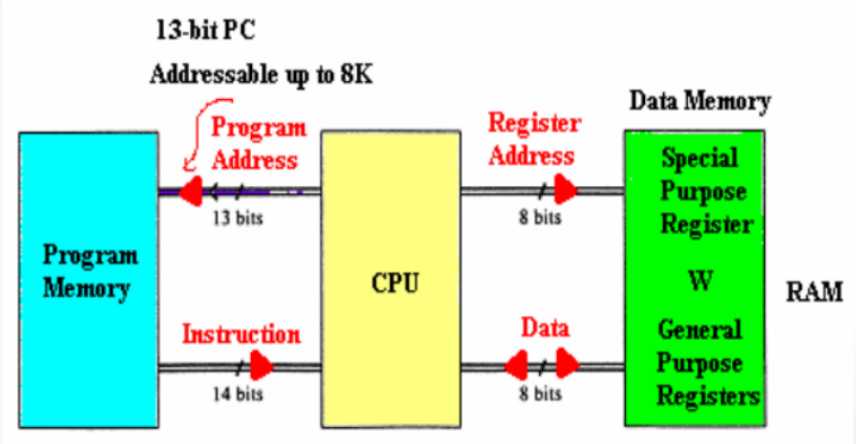



# PIC mikrodenetleyicileri

Tüm mikrodenetleyiciler 8-bit veri yoluna sahiptir fakat program uzunlukları değişir.

| | Baseline Architecture | Mid-Range Architecture | Enhanced Mid-Range Architecture | PIC18 Architecture |
|---|---|---|---|---|
| Pin Count | 6-40 | 8-64 | 8-64 | 18-100 |
| Interrupts | No | Single interrupt capability | Single interrupt capability with hardware context save | Multiple interrupt capability with hardware context save |
| Performance | 5 MIPS | 5 MIPS | 8 MIPS | Up to 16 MIPS |
| Instructions | 33, 12-bit | 35, 14-bit | 49, 14-bit | 83, 16-bit |
| Program Memory | Up to 3 KB | Up to 14 KB | Up to 28 KB | Up to 128 KB |
| Data Memory | Up to 138 Bytes | Up to 368 Bytes | Up to 1,5 KB | Up to 4 KB |
| Hardware Stack | 2 level | 8 level | 16 level | 32 level |
| Features | ▪ Comparator<br>▪ 8-bit ADC<br>▪ Data Memory<br>▪ Internal Oscillator | In addition to Baseline:<br>▪ SPI/I²C™<br>▪ UART<br>▪ PWMs<br>▪ LCD<br>▪ 10-bit ADC<br>▪ Op Amp | In addition to Mid-Range:<br>▪ Multiple Communication Peripherals<br>▪ Linear Programming Space<br>▪ PWMs with Independent Time Base | In addition to Enhanced Mid-Range:<br>▪ 8x8 Hardware Multiplier<br>▪ CAN<br>▪ CTMU<br>▪ USB<br>▪ Ethernet<br>▪ 12-bit ADC |
| Highlights | Lowest cost in the smallest form factor | Optimal cost to performance ratio | Cost effective with more performance and memory | High performance, optimized for C programming, advanced peripherals |




Örnek: PIC16F877A

# PIC mikrodenetleyicileri

- Low End: 12C508
  - 8pin package (DIP)
  - 12bit core – 33 instruction
  - 1us instruction time (Tclk=4MHz)
  - 25 8bit data memory or register
  - 2 level hardware stack (no interrupts)
  - 5 GPIO pins, 1 input only (25mA sink/source)
  - Internal pull-up, wake up on pin change, internal oscillator, Timer, WDT

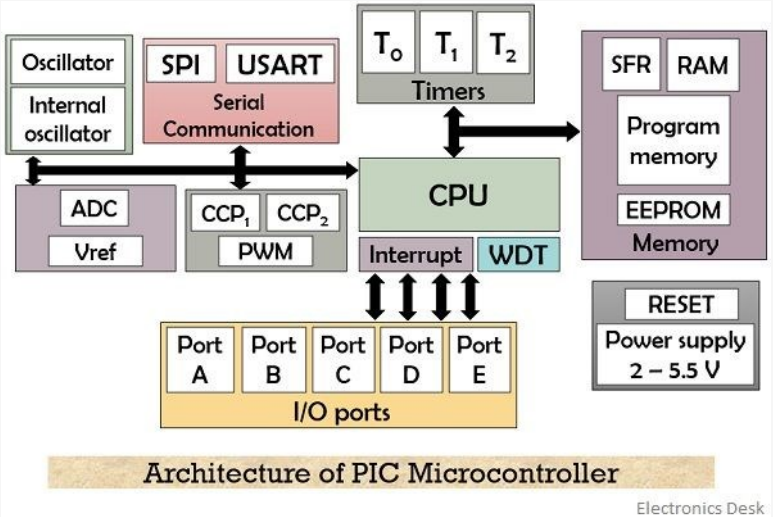
Mid-range: 16F876

- 28pin package (DIP)
- 14bit core – 35 instruction
- 200ns instruction time (Tclk=20MHz)
- 368 8bit data memory or register
- 8 level hardware stack (no interrupts)
- 22 GPIO pins, (20mA source, 25mA sink)
- 5ch 10bit ADC, USART/I2C/SPI, 16 bit / 8 bit timers, Brown out detect, ICD.

# PIC mikrodenetleyici avantaj/dezavantajları

*Avantaj:*

- Piyasada en kolay bulunabilecek mikrodenetleyicilerdendir.
- En ucuz ve en basit mikroişlemciler arasında yer alır.
- Harvard mimarisi kullandığından veriyolları farklı genişliklerdedir.
- Sabit kesme gecikmesi sağlar.




*Dezavantaj:*

- Tek akümülatör (accumulator) ve küçük bir komut seti var.
- Program hafızasına doğrudan ulaşılamaz.
- Adresleme modları oldukça az çoğunlukla hafızaya doğrudan erişerek yapılır.

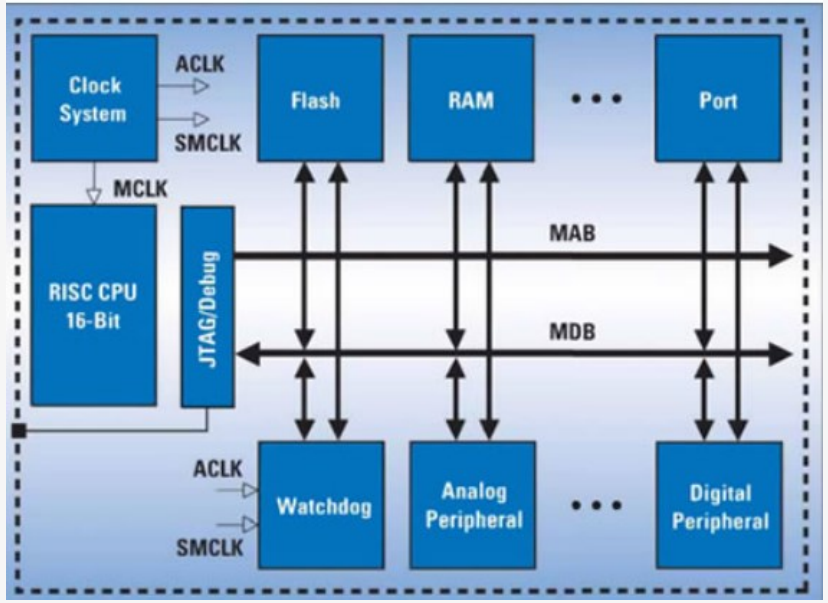
# MSP430 mikrodenetleyicileri (16-bit)

- MSP mikrodenetleyiciler, Texas Instruments firmasına aittir.

Genel Özellikleri:

- Düşük ücretlidirler ve düşük güç tüketim modlarında çok düşük güç tüketirler.
- Von Neumann mimarisindedir.
- 27 temel ve 24 türetilmiş komut setine sahiptir (RISC).
- 16 bit veri yolu sahiptirler.
- 16 tane genel amaçlı register barındırırlar.
- 7 farklı adresleme modu içerirler (RISC).
- Hafızadan hafızaya veri transferi yapabilirler.
- Birden fazla clock seçeneği (MCLK, ACLK).

MSP430 architecture.



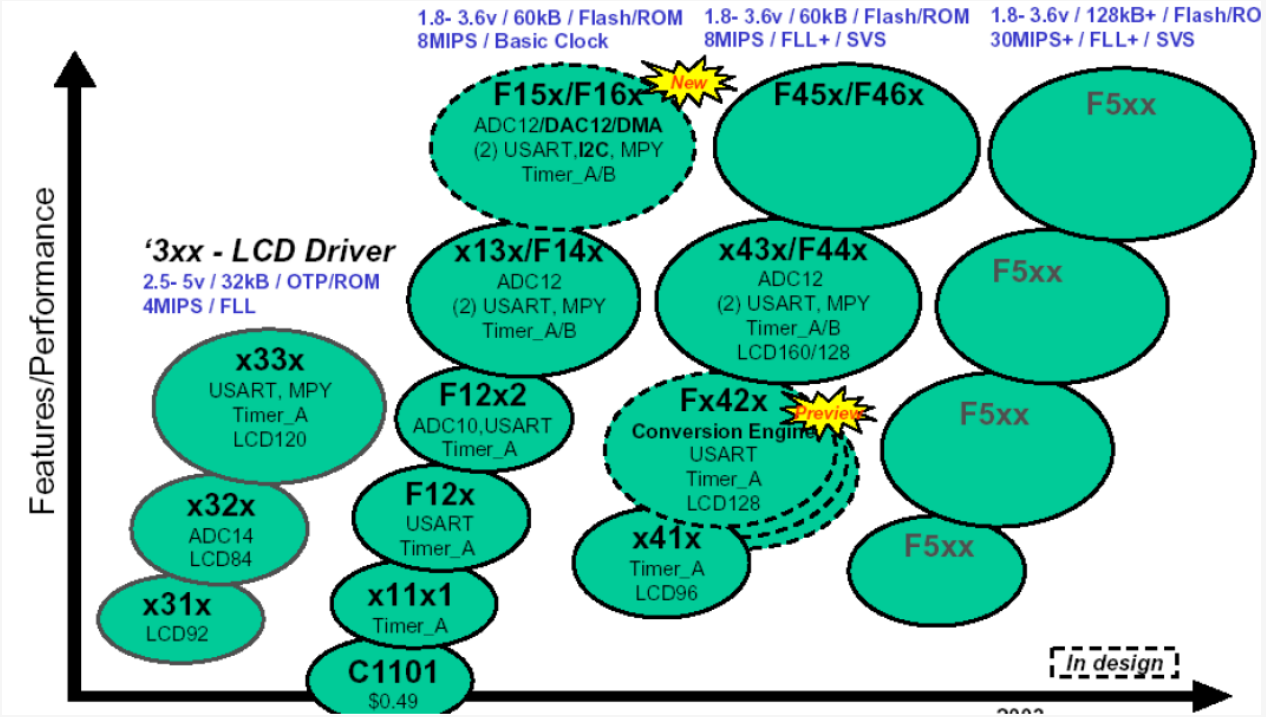
# MSP430 mikrodenetleyicileri (16-bit)

## Broad family of TI's 16-bit microcontrollers

- from 1Kbytes ROM, 128 bytes RAM (approx. $1 )
- to 60Kbytes ROM, 10Kbytes RAM ( $10)

## Three subfamilies

- MSP430x1xx: basic unit
- MSP430x3xx: more features
- MSP430x4xx: built-in LCD driver




Credit by Dr. Aleksander Milenkovic

# MSP430 mikrodenetleyicileri (16-bit) Komut Seti

| | | | | Status Bits | | | |
|---|---|---|---|---|---|---|---|
| | | | | V | N | Z | C |
| * | ADC(.B) | dst | dst + C → dst | x | x | x | x |
| | ADD(.B) | src,dst | src + dst → dst | x | x | x | x |
| | ADDC(.B) | src,dst | src + dst + C → dst | x | x | x | x |
| | AND(.B) | src,dst | src .and. dst → dst | 0 | x | x | x |
| | BIC(.B) | src,dst | .not.src .and. dst → dst | - | - | - | - |
| | BIS(.B) | src,dst | src .or. dst → dst | - | - | - | - |
| | BIT(.B) | src,dst | src .and. dst | 0 | x | x | x |
| * | BR | dst | Branch to ..... | - | - | - | - |
| | CALL | dst | PC+2 → stack, dst → PC | - | - | - | - |
| * | CLR(.B) | dst | Clear destination | - | - | - | - |
| * | CLRC | | Clear carry bit | - | - | - | 0 |
| * | CLRN | | Clear negative bit | - | 0 | - | - |
| * | CLRZ | | Clear zero bit | - | - | 0 | - |
| | CMP(.B) | src,dst | dst - src | x | x | x | x |
| * | DADC(.B) | dst | dst + C → dst (decimal) | x | x | x | x |
| | DADD(.B) | src,dst | src + dst + C → dst (decimal) | x | x | x | x |
| * | DEC(.B) | dst | dst - 1 → dst | x | x | x | x |
| * | DECD(.B) | dst | dst - 2 → dst | x | x | x | x |
| * | DINT | | Disable interrupt | - | - | - | - |
| * | EINT | | Enable interrupt | - | - | - | - |
| * | INC(.B) | dst | Increment destination, dst +1 → dst | x | x | x | x |
| * | INCD(.B) | dst | Double-Increment destination, dst+2→dst | x | x | x | x |
| * | INV(.B) | dst | Invert destination | x | x | x | x |
| | JC/JHS | Label | Jump to Label if Carry-bit is set | - | - | - | - |
| | JEQ/JZ | Label | Jump to Label if Zero-bit is set | - | - | - | - |
| | JGE | Label | Jump to Label if (N .XOR. V) = 0 | - | - | - | - |
| | JL | Label | Jump to Label if (N .XOR. V) = 1 | - | - | - | - |
| | JMP | Label | Jump to Label unconditionally | - | - | - | - |
| | JN | Label | Jump to Label if Negative-bit is set | - | - | - | - |

Legend: 0 Status bit always cleared; 1 Status bit always set; x Status bit cleared or set on results; - Status bit not affected; * Emulated Instructions

| | | | | Status Bits | | | |
|---|---|---|---|---|---|---|---|
| | | | | V | N | Z | C |
| | JNC/JLO | Label | Jump to Label if Carry-bit is reset | - | - | - | - |
| | JNE/JNZ | Label | Jump to Label if Zero-bit is reset | - | - | - | - |
| | MOV(.B) | src,dst | src → dst | - | - | - | - |
| * | NOP | | No operation | - | - | - | - |
| * | POP(.B) | dst | Item from stack, SP+2 → SP | - | - | - | - |
| | PUSH(.B) | src | SP - 2 → SP, src → @SP | - | - | - | - |
| | RETI | | Return from interrupt<br>TOS → SR, SP + 2 → SP<br>TOS → PC, SP + 2 → SZP | x | x | x | x |
| * | RET | | Return from subroutine<br>TOS → PC, SP + 2 → SP | - | - | - | - |
| * | RLA(.B) | dst | Rotate left arithmetically | x | x | x | x |
| * | RLC(.B) | dst | Rotate left through carry | x | x | x | x |
| | RRA(.B) | dst | MSB → MSB ....LSB → C | 0 | x | x | x |
| | RRC(.B) | dst | C → MSB .....LSB → C | x | x | x | x |
| * | SBC(.B) | dst | Subtract carry from destination | x | x | x | x |
| * | SETC | | Set carry bit | - | - | - | 1 |
| * | SETN | | Set negative bit | - | 1 | - | - |
| * | SETZ | | Set zero bit | - | - | 1 | - |
| | SUB(.B) | src,dst | dst + .not.src + 1 → dst | x | x | x | x |
| | SUBC(.B) | src,dst | dst + .not.src + C → dst | x | x | x | x |
| | SWPB | dst | swap bytes | - | - | - | - |
| | SXT | dst | Bit7 → Bit8 ..... Bit15 | 0 | x | x | x |
| * | TST(.B) | dst | Test destination | x | x | x | x |
| | XOR(.B) | src,dst | src .xor. dst → dst | x | x | x | x |

Legend: 0 The Status Bit is cleared; 1 The Status Bit is set; x The Status Bit is affected; - The Status Bit is not affected; * Emulated Instructions

# MSP430 mikrodenetleyicileri (16-bit)

**Table 4.3** MSP430 addressing modes

| Addressing mode | Syntax | Comment |
|---|---|---|
| Immediate mode | #X | Data is X |
| Register Mode | Rn | Data is Register Rn Contents |
| *Data in Memory:* | | |
| Indexed mode | X(Rn) | Address is X + Rn contents |
| Indirect mode | @Rn | Address is Register Rn contents |
| Indirect autoincrement | @Rn+ | Address is Register Rn contents as before. Now, Rn is incremented after execution by 2 for word instructions and by 1 for byte ones. |
| Direct mode | X | Address is X |
| Absolute mode | &X | Address is X |

# MSP430 düşük güç tüketim modları

- **One of the main features of the MSP430 families:**
  - Low power consumption (about 1 mW/MIPS or less);
  - Important in battery operated embedded systems.

- **Low power consumption is only accomplished:**
  - Using low power operating modes design;
  - Depends on several factors such as:
    - Clock frequency;
    - Ambient temperature;
    - Supply voltage;
    - Peripheral selection;
    - Input/output usage;
    - Memory type;
    - ...

# MSP430 düşük güç tüketim modları

- **Low power modes (LPM):**
  - 6 operating modes;
  - Configured by the SR bits: CPUOFF, OSCOFF, SCG1, SCG0.

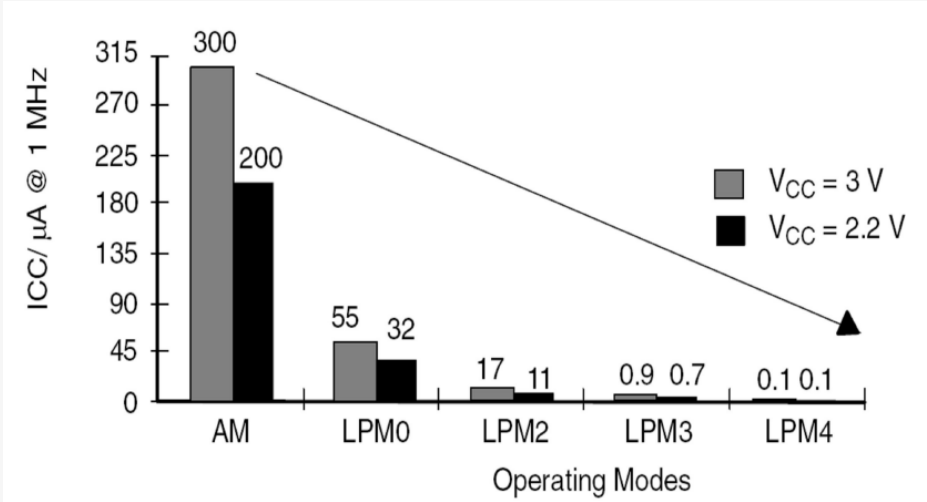
  - **Active mode (AM) - highest power consumption:**
    - Configured by disabling the SR bits described above;
    - CPU is active;
    - All enabled clocks are active;
    - Current consumption: 250 μA.

  - **Software selection up to 5 LPM of operation;**

  - **Operation:**
    - An interrupt event can wake up the CPU from any LPM;
    - Service the interrupt request;
    - Restore back to the LPM.

# MSP430 düşük güç tüketim modları

- **Low power modes (LPM) characteristics:**
  - **LPM0 to LPM3:**
    - Periodic processing based on a timer interrupt;
    - **LPM0**: CPU and MCLK are disabled, SMCLK and ACLK remain active. Both DCO source signal and DCO's DC gen. are active. This is used when the CPU is not required but some modules require a fast clock from SMCLK and the DCO
    - **LPM1**: Main difference to LPM0 is the condition of enable/disable the DCO's DC generator;
    - **LPM2**: DCO's DC generator is active and DCO is disabled;
    - **LPM3**: CPU, MCLK, SMCLK, and DCO are disabled; only ACLK remains active (< 2 μA). This is the standard low-power mode when the device must wake itself at regular intervals and therefore needs a (slow) clock. It is also required if the MSP430 must maintain a real-time clock.
  - **LPM4:**
    - The device can be wakened only by an external signal.
    - No clocks are active and available for peripherals.
    - Reduced current consumption (0.1 μA).



# Örnek Düşük Güç Tüketimi Uygulaması

Butona basıldığında kesmeyi aktif hale getir ve kesme içerisinde ledin durumunu tersle. Diğer tüm durumlarda işlemciyi düşük güç moduna al.

```
// butled4.c - press button B1 to light LED1
// Responds to interrupts on input pin , LPM4 between interrupts
// Olimex 1121 STK board , LED1 active low on P2.3,
// button B1 active low on P2.1
// J H Davies , 2006 -11 -18; IAR Kickstart version 3.41A
// ------------------------------------------------------------------
#include <io430x11x1.h> // Specific device
#include <intrinsics.h> // Intrinsic functions
// ------------------------------------------------------------------
void main (void)
{
    WDTCTL = WDTPW | WDTHOLD;          // Stop watchdog timer
    P2OUT_bit.P2OUT_3 = 1;             // Preload LED1 off (active low!)
    P2DIR_bit.P2DIR_3 = 1;             // Set pin with LED1 to output
    P2IE_bit.P2IE_1 = 1;               // Enable interrupts on edge
    P2IES_bit.P2IES_1 = 1;             // Sensitive to negative edge (H->L)
    do {
        P2IFG = 0;                     // Clear any pending interrupts ...
    } while (P2IFG != 0);              // ... until none remain
    for (;;) {                         // Loop forever (should not need)
    __low_power_mode_4 ();             // LPM4 with int'pts , all clocks off
    }                                  // (RAM retention mode)
}
// ------------------------------------------------------------------
```

```
// Interrupt service routine for port 2 inputs
// Only one bit is active so no need to check which
// Toggle LED , toggle edge sensitivity , clear any pending
   interrupts
// Device returns to low power mode automatically after ISR
// ------------------------------------------------------------------
#pragma vector = PORT2_VECTOR
__interrupt void PORT2_ISR (void)
{
   P2OUT_bit.P2OUT_3 ^= 1; // Toggle LED
   P2IES_bit.P2IES_1 ^= 1; // Toggle edge sensitivity
   do {
       P2IFG = 0; // Clear any pending interrupts ...
   } while (P2IFG != 0); // ... until none remain
}
```