MİKROİŞLEMCİLER (BLM202)

HAFTA - 4

Dr. Bilgin YAZLIK, RTTP, PMP

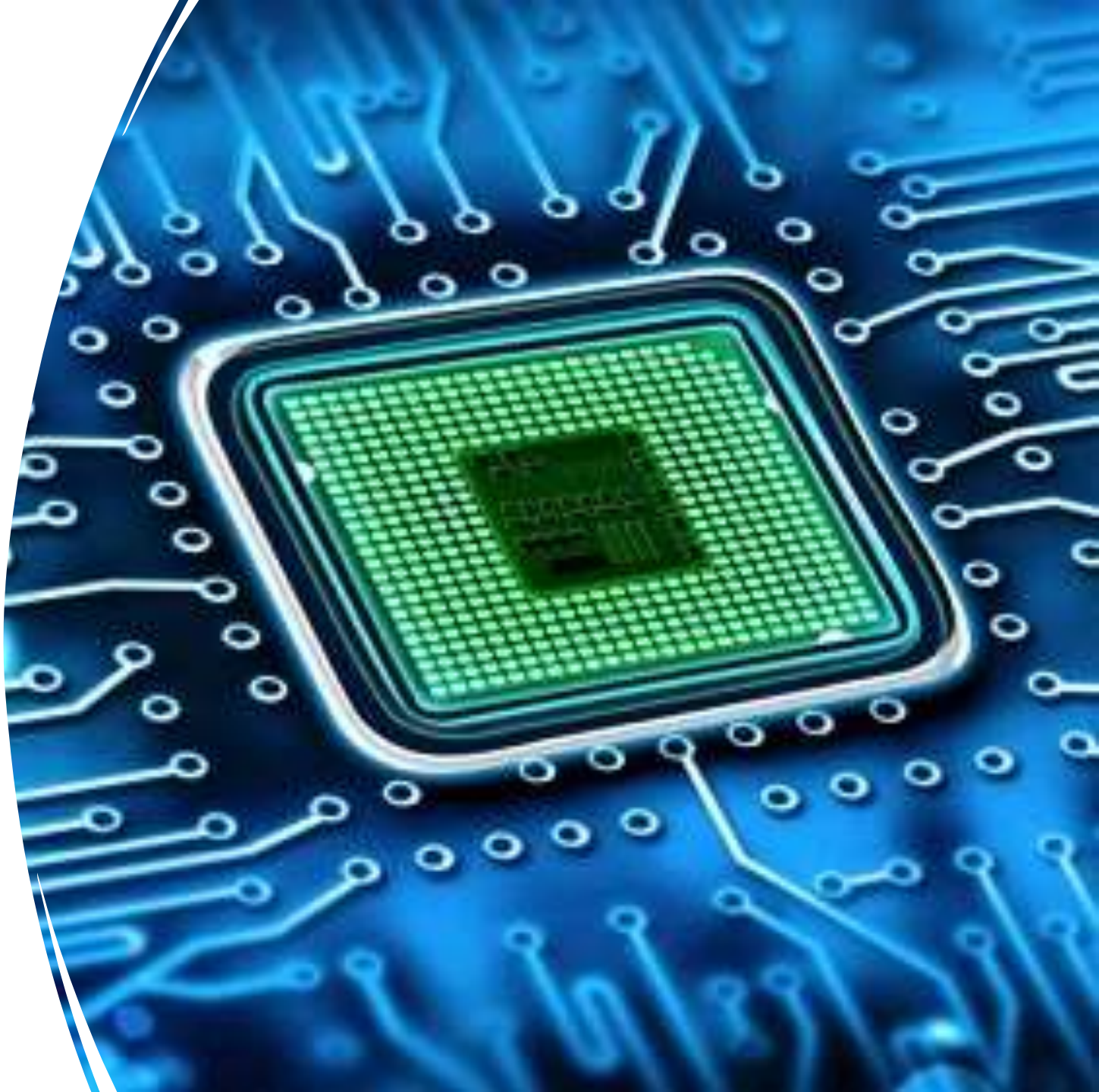
NEVŞEHİR HACI BEKTAŞ VELİ ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

- 2209-A Programı 2023 yılı başvuruları 25 Nisan-26 Mayıs 2023 tarihleri arasında alınacaktır. Çağrı duyurusu daha sonra açıklanacaktır.

4. HAFTA

- CISC ve RISC Mimarileri
- Assembly Giriş



CISC VE RISC MİMARİSİ

CISC ve RISC Mimarileri Nedir?

Mikroişlemcinin temel unsurları kaydediciler, veri yolları ve iş hatlarıdır. Bu unsurların büyüklüğü, sayısı, yapısı o işlemcinin yeteneklerini belirler ve bir mimariyi diğer mimarilerden ayırır.

Bilgisayar tarihinin başlarında, donanım fiyatlarının yüksek oluşundan dolayı çoğu bilgisayar oldukça basit komut kümesine sahipti. Sonraki yıllarda donanımı oluşturan elemanların üretimindeki artış, fiyatların düşmesine bunun sonucunda sistemde yüksek miktarda eleman kullanılmasına sebep oldu. Böylece fazla donanım kullanımı, komut kümesinin büyümesini ve sistemi çok karmaşık yapan donanımlarda kullanılmasını sağlamıştır.

Bir bilgisayarın komut kümesi, programcının makineyi programlarken kullanabileceği ilkel emirleri veya makine komutlarının tamamının oluşturduğu kümeyi belirtir. Bir komut setinin karmaşıklığı, komut ve veri formatlarına, adresleme modlarına, genel amaçlı kaydedicilere, opcode tanımlamalarına ve kullanılan akış kontrol mekanizmalarına bağlıdır. İşlemci tasarımındaki komut seti mimarileri CISC ve RISC olmak üzere iki çeşittir.

CISC ve RISC Mimarileri Nedir?

- Hesaplama hızı, hassaslık ve programlama kolaylığı sağlamak için değişik işlemlerle zenginleştirerek değişik işlemciler tasarlamak mümkündür.
- En yaygın olan mimariler
- **CISC** (Complex Instruction Set Computer – Komut Kümesi Karmaşık Olan Bilgisayar) ve
- **RISC** (Reduced Instruction Set Computer – Komut Kümesi Asgari Olan Bilgisayar) mimarileridir.

CISC NEDİR? (Complex Instruction Set Computer)

Intel'in X86 mimarisine dayalı işlemci serisinin ortaya çıktığı 70'li yıllarda, RAM'lerin pahalı ve kısıtlı olması sebebiyle bu kaynakların tasarruflu bir şekilde kullanılarak yüksek seviyeli dillerin desteklenmesini savunan bazı tasarım mimarları bir araya gelerek CISC mimarisini geliştirmişlerdir. Bu mimari, programlanması kolay ve etkin bellek kullanımı sağlayan tasarım felsefesinin bir ürünüdür. Her ne kadar performans düşüklüğüne sahip olsa ve işlemciyi karmaşık hale getirse de yazılımı basitleştirmektedir.

CISC mimarisinin karakteristik iki özelliğinden birisi, değişken uzunluktaki komutlar, diğeri ise karmaşık komutlardır. Değişken ve karmaşık uzunluktaki komutlar bellek tasarrufu sağlar. Karmaşık komutlar iki ya da daha fazla komutu tek bir komut haline getirdikleri için hem bellekten hem de programda yer alması gereken komut sayısından tasarruf sağlar. Karmaşık komut karmaşık mimariyi de beraberinde getirir. Mimarideki karmaşıklığın artması, işlemci performansında istenmeyen durumların ortaya çıkmasına sebep olur. Ancak programların yüklenmesinde ve çalıştırılmasındaki düşük bellek kullanımı bu sorunu ortadan kaldırabilir.

CISC

CISC mimarisi çok kademeli işleme modeline dayanmaktadır. İlk kademe yüksek düzeyli dilin yazıldığı yerdir. Sonraki kademeyi makine dili oluşturur ki, yüksek düzeyli dilin derlenmesi sonucu bir dizi komutlar makine diline çevrilir. Bir sonraki kademede makine diline çevrilen komutların kodları çözülerek, mikroişlemcinin donanım birimlerini kontrol edebilen en basit işlenebilir kodlara (mikrokod) dönüştürülür. En alt kademede ise işlenebilir kodları alan donanım aracılığıyla gerekli görevler yerine getirilir.

RISC NEDİR? (REDUCED INSTRUCTION SET COMPUTER)

RISC mimarisi, CISC mimarili işlemcilerin kötü yanlarını gidermek için piyasanın tepkisi ile ona bir alternatif olarak geliştirilmiştir. RISC'ı IBM, Apple ve Motorola gibi firmalar sistematik bir şekilde geliştirmiştir. RISC felsefesinin taraftarları, bilgisayar mimarisinin tam anlamıyla bir elden geçirmeye ihtiyacı olduğunu ve neredeyse bütün geleneksel bilgisayarların mimari bakımından birtakım eksikliklere sahip olduğunu ve eskidiğini düşünüyorlardı. Bilgisayarların gittikçe daha karmaşık hale getirildiği ve hepsinin bir kenara bırakılıp en baştan geri başlamak gerektiği fikrindeydiler.

70'lerin ortalarında yarı iletken teknolojisindeki gelişmeler, ana bellek ve işlemci yongaları arasındaki hız farkını azaltmaya başladı. Bellek hızı artırıldığından ve yüksek seviyeli diller Assembly dilinin yerini aldığından, CISC'ın başlıca üstünlükleri geçersizleşmeye başladı. Bilgisayar tasarımcıları sadece donanımı hızlandırmaktan çok bilgisayar performansını iyileştirmek için başka yollar denemeye başladılar.

RISC NEDİR?

RISC mimarisi aynı anda birden fazla komutun birden fazla birimde işlendiği *iş hatlı tekniği ve süperskalar* yapılarının kullanımıyla yüksek bir performans sağlamıştır. Bu tasarım tekniği yüksek bellek ve gelişmiş derleme teknolojisi gerektirmektedir. Bu mimari küçültülen komut kümesi ve azaltılan adresleme modları sayısı yanında aşağıdaki özelliklere sahiptir.

- Bir çevrimlik zamanda bir komut işleyebilme
- Aynı uzunluk ve sabit form atta komut kümesine sahip olma
- Ana belleğe sadece load ve store komutlarıyla erişim, operasyonların sadece kaydedici üzerinde yapılması
- Bütün icra birimlerinin mikrokod kullanılmadan donanımsal çalışması
- Yüksek seviyeli dilleri destekleme
- Çok sayıda kaydediciye sahip olması

MIPS R4000

MIPS III ISA (64-bit)

Released 1991
100 Mhz
1.35 million transistors
133 instructions

RISC

Intel i486

80486 ISA (32-bit)

Released 1989
20-50 Mhz (1991)
1.18 million transistors
180 instructions (estimate)

CISC

PowerPC 601

PowerPC ISA (32-bit)

Released 1993
50 - 80 Mhz
2.8 million transistors
273 instructions

RISC

Intel Pentium

Pentium ISA (32-bit)

Released 1993
60 - 66 Mhz
3.1 million transistors
221 instructions

CISC

Comparison of transistors and instruction count for popular RISC and CISC processors in the early 1990s

CISC ve RISC Mimarileri Nedir?

- Adı geçenler en yaygın olan mimarilerdir. Bunların dışında, basit bir hesap makinesinden başlamış, en karmaşık süper bilgisayarlara kadar ne varsa her birinin orijinal bir mimarisi olmasa bile, en azından bir mimari özelliği vardır.
- **$T_p = S \times A \times T_a$**
- T_p= Programın icra süresi
- S: Komutların Sayısı
- A: Bir komutun sürüldüğü aşamaların ortalama sayısı
- T_a: Bir aşamayı sürdürme süresi

CISC ve RISC Mimarileri Nedir?

- $T_p = S \times A \times T_a$
- T_p nin en minimum değeri doğal olarak S , A ve T_a'nın minimum değerlerinde elde edilir.
- Ancak takdir edersiniz ki bu 3 parametrenin aynı anda minimum olma şansı yoktur.
- Bilgisayarın yapı ve çalışma özelliğinden dolayı buradaki bir parametre düşerken diğeri yükselmekte, diğeri düşerken başka bir parametre yükselmektedir.
- Burada CISC ve RISC mimarileri iki ayrı yöntem kullanmıştır.

CISC Mimarili İşlemciler

- T_p'yi en aza indirmek için, CISC mimari, S'yi (komut sayısı) minimumlaştırmak yolunu seçmiş ve bu sırada A ve T_a değerlerinin yükselmesinden doğacak kayıpların kazanca nispeten daha az olacağını iddia etmektedir.
- Uygulama alanında yoğun olarak rastlanması tahmin edilen ve nispeten karmaşık olan işlemler içinde komutlara sahiptir. Bu komutlar asembler dilinde kolay ve hızlı programlama ortamı sağlarlar. Yüksek seviyeli programlama dillerini destekler ve yüksek performanslı derleyiciler tasarlamak gerektiğini iddia eder.

CISC Mimarili İşlemciler

- Örnek:
  C'yi 1 azalt
  Eğer C =! 0 ise döngü başına atla
- gibi temel işlemler sırasıyla DEC ve JNZ komutlarıyla gerçekleştirilir.
- Fakat programlamalarda bu iki işlem çok yoğun olarak birlikte rastlandığından dolayı 8086 dan başlayarak Intel'in bütün mikroişlemcileri bu iki komutu birlikte icra eden LOOPNZ komutu ile donatılmaktadır.
- Bunun gibi bir çok örnek vardır.

CISC Dezavantajı

- Karmaşık işlemleri gerçekleştiren komutlar programlarda seyrek olarak rastlanmakta ve assembler dilinde programlamayı kolaylaştırmanın bedeli olarak derleyicilerin işini zorlaştırmaktadır.
- Yani ayrı ayrı problemlerin assemblerda kolay programlanması açısından faydalı olan karmaşık komutlar, yüksek seviyeli dillerde yazılmış programların kolay ve hızlı derlenmesi yolunda bir kargaşaya sebep olurlar.
- Bu dezavantaj, bir grup işlemci üreticisinin alternatif olan RISC mimarisine yönelmesi için esas sebep olmuştur.

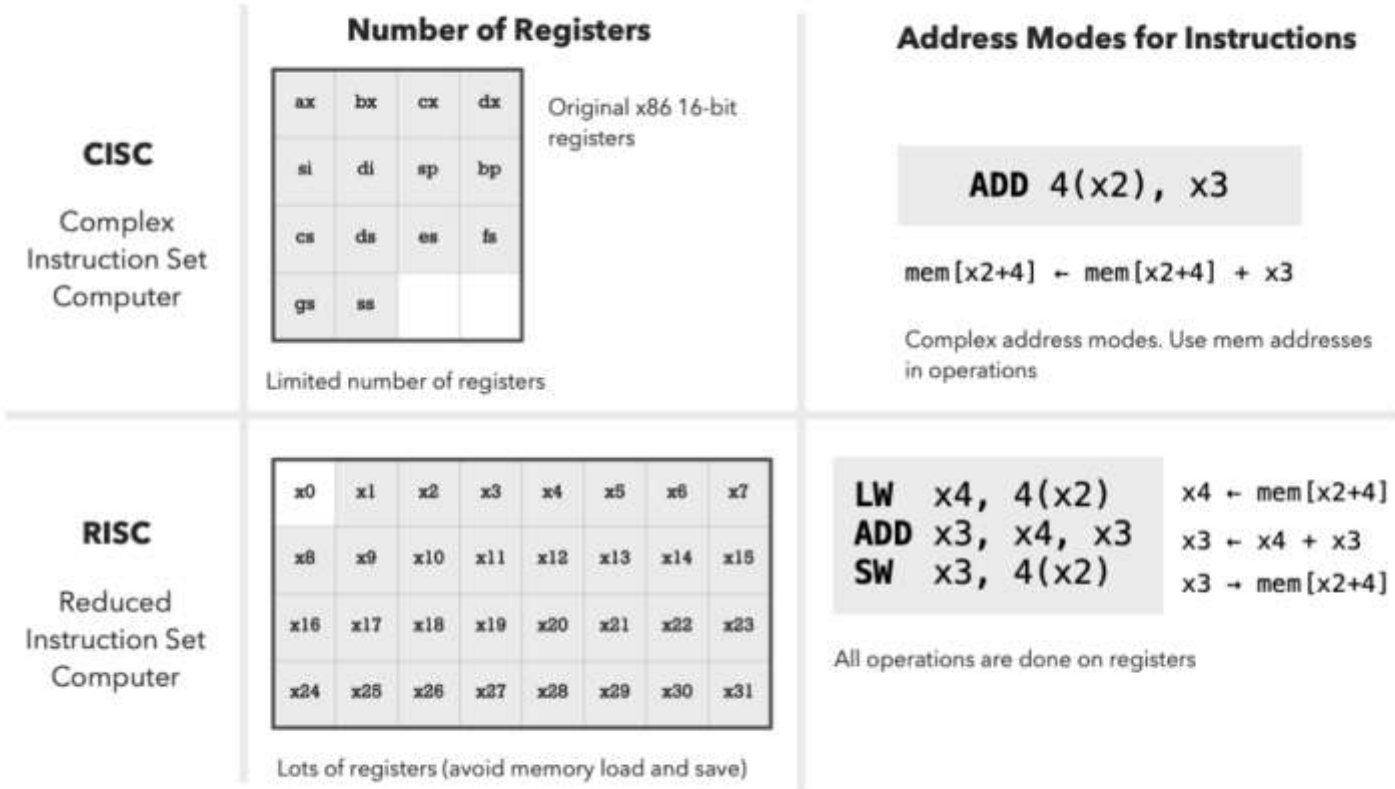
RISC Mimarili İşlemciler

- RISC mimarisinde, A x T_a (Aşama Sayısı x Aşama Süresi) çarpımını minimumlaştırmak yolu tercih edilmiştir, bu sırada S'nin değerinin yükselmesinden doğacak kayıpların kazanca nispeten az olacağını iddia edilmektedir.
- Hatırlatalım ki bunlar alternatif gibi görünse de birbirini reddeden mimariler değil, hatta bir sistem dahilinde bir araya gelebilen mimarilerdir.
- CISC mikroişlemciler tarafından her biri tek bir çok fonksiyonlu komutla gerçekleştirilebilen işlemler RISC mikroişlemciler tarafından birkaç komutla gerçekleştirilir. Derleyici için kolay olmasına ve basit donanımla hızla gerçekleştirilmesine özellikle dikkat edilir.

RISC Mimarili İşlemciler

- Aynı problem için tipik bir RISC ve tipik bir CISC işlemcisine yönelik olarak tertiplenmiş olan iki program karşılaştırıldığında, kullandıkları komutların toplam sayısına göre RISC mimarili program ikinciden ortalama olarak 1.66 kat uzun olur.
- Burada RISC mimari yapısının verimlilik kaybı olduğu kanaatine varılabilir, bu teorik olarak doğru gibi gözükse de gerçekte böyle değildir. Çünkü RISC mimarisi, söz konusu kayıpları kısmen önlemek, kısmen de telafi etmek için kendine özgü ilkeler bulmuştur.
- Bir çok özel amaçlı kontrol kartlarının merkezinde RISC işlemcileri durur. Bunlardan en yaygın olanı : PIC 8086, IBM 801, MIPS ailesi, SPRAS ailesi, POWER PC ailesi, Motorola 88000 ailesi, ARM vb.

RISC VE CISC FARKLARI



CISC	RISC
Emphasis on hardware	Emphasis on software
Multiple instruction sizes and formats	Instructions of same set with few formats
Less registers	Uses more registers
More addressing modes	Fewer addressing modes
Extensive use of microprogramming	Complexity in compiler
Instructions take a varying amount of cycle time	Instructions take one cycle time
Pipelining is difficult	Pipelining is easy

KOMUT SETİ

Intel Pentium Instruction Set Reference

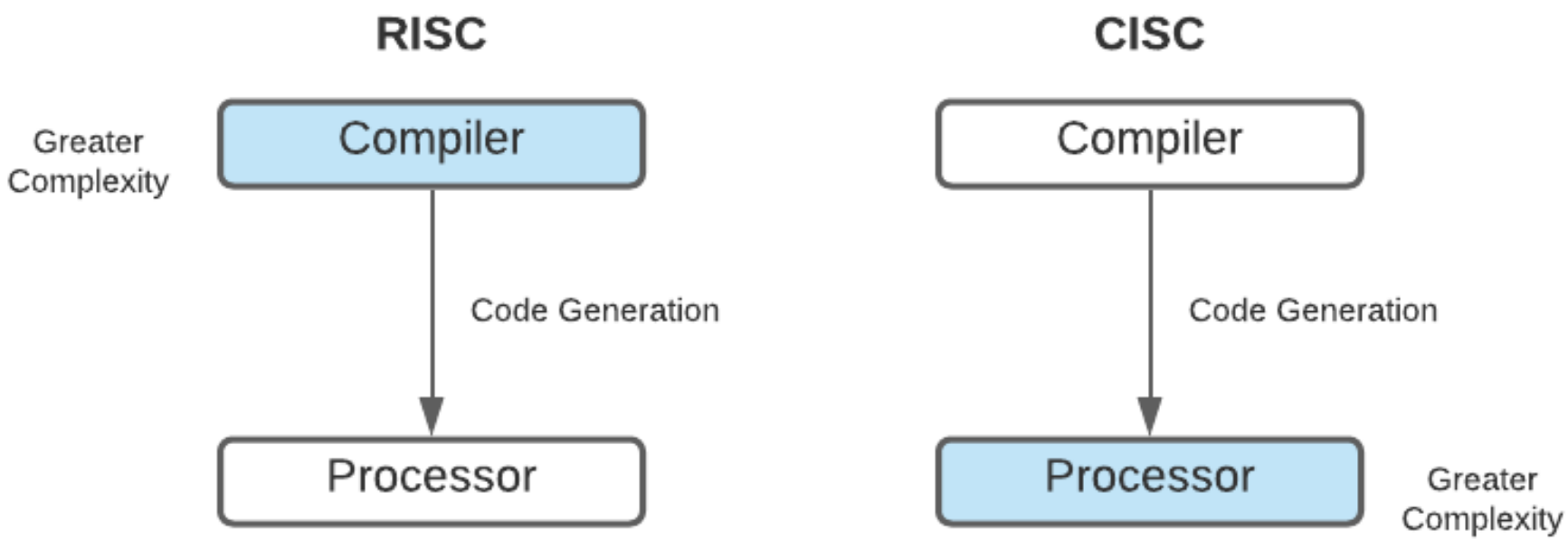
Instruction Index

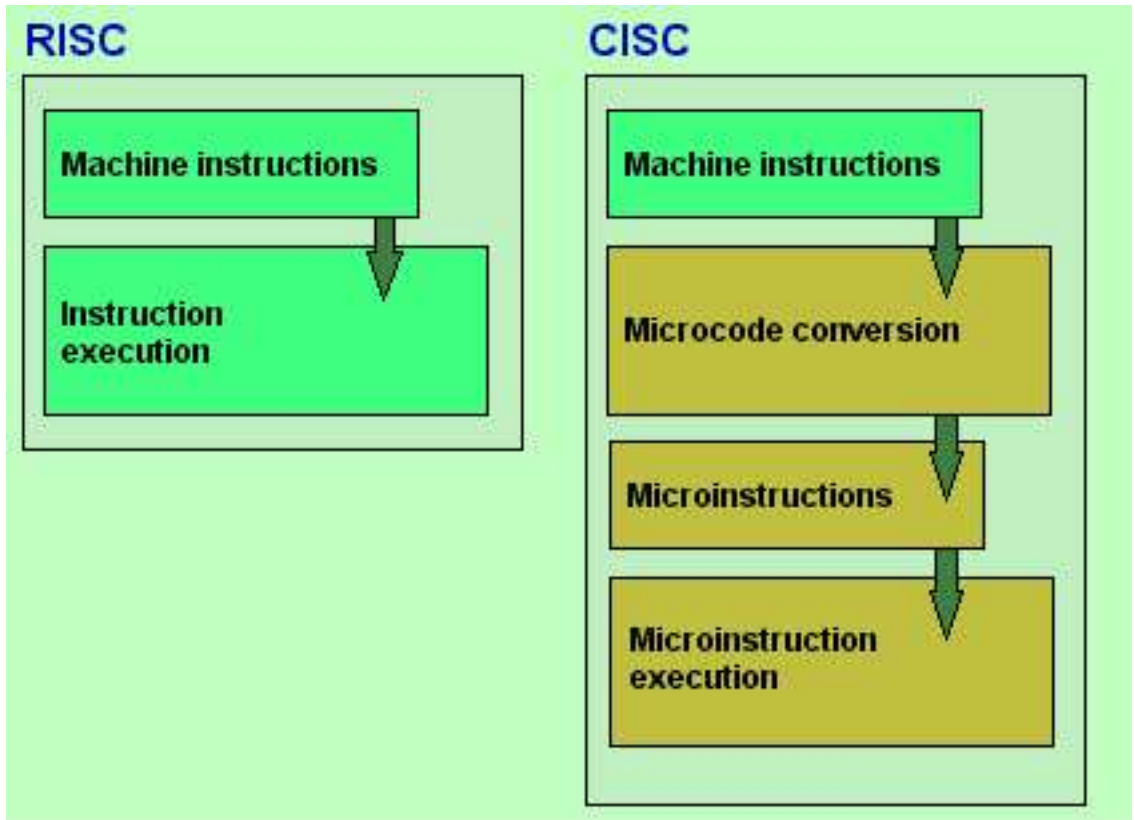
A

- **AAA** - ASCII Adjust for Addition
- **AAD** - ASCII Adjust AX Before Division
- **AAM** - ASCII Adjust AX After Multiply
- **AAS** - ASCII Adjust AL After Subtraction
- **ADC** - Add with Carry
- **ADD** - Add
- **AND** - Logical AND
- **ARPL** - Adjust RPL Field of Segment Selector

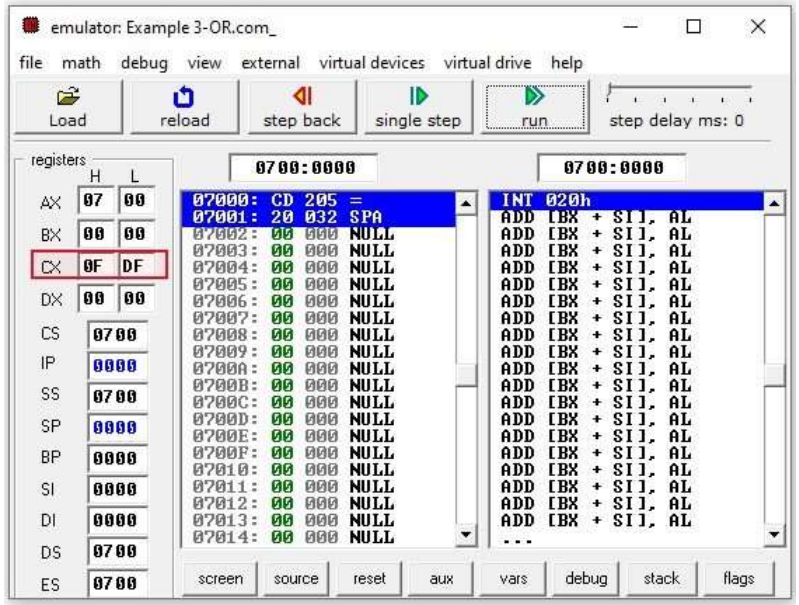
B

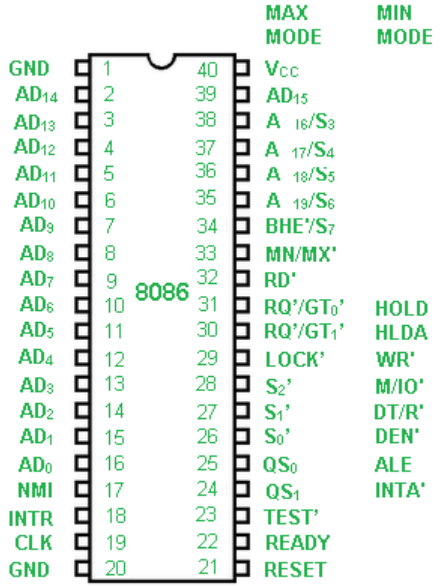
- **BOUND** - Check Array Index Against Bounds
- **BSF** - Bit Scan Forward
- **BSR** - Bit Scan Reverse
- **BSWAP** - Byte Swap
- **BT** - Bit Test
- **BTC** - Bit Test and Compliment
- **BTR** - Bit Test and Reset
- **BTS** - Bit Test and Set

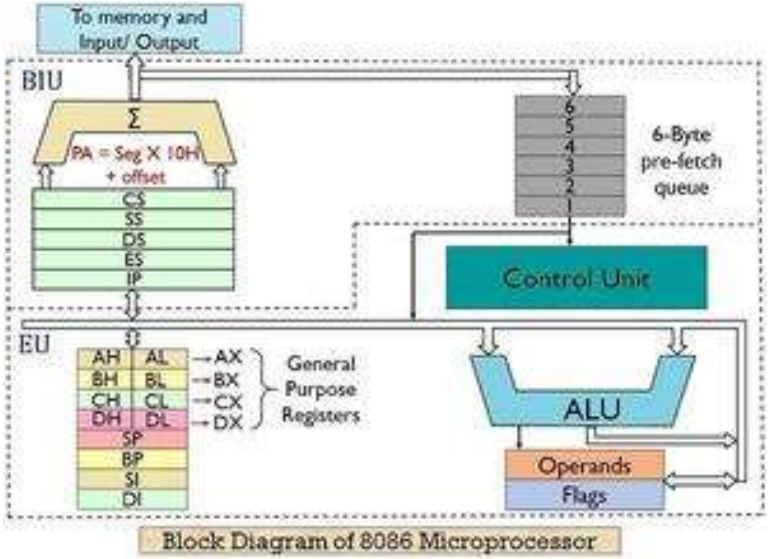

RISC
Greater Complexity
Compiler
Code Generation
Processor
CISC
Compiler
Code Generation
Processor
Greater Complexity


RISC
Machine instructions
Instruction execution
CISC
Machine instructions
Microcode conversion
Microinstructions
Microinstruction execution


Assembly
Language
C
C++
ASSEMBLER
DELPHI
PROGRAMMING
PASCAL
PYTHON
ACTIONSCRIPT
PYTHON
PERL
JAVASCRIPT
PROGRAMMING
PYTHON
MYSQL
PROGRAMMING
JAVA
ASSEMBLER
XML
PYTHON
PHP
RUBY
JAVASCRIPT
PROGRAMMING
PERL
0101010101







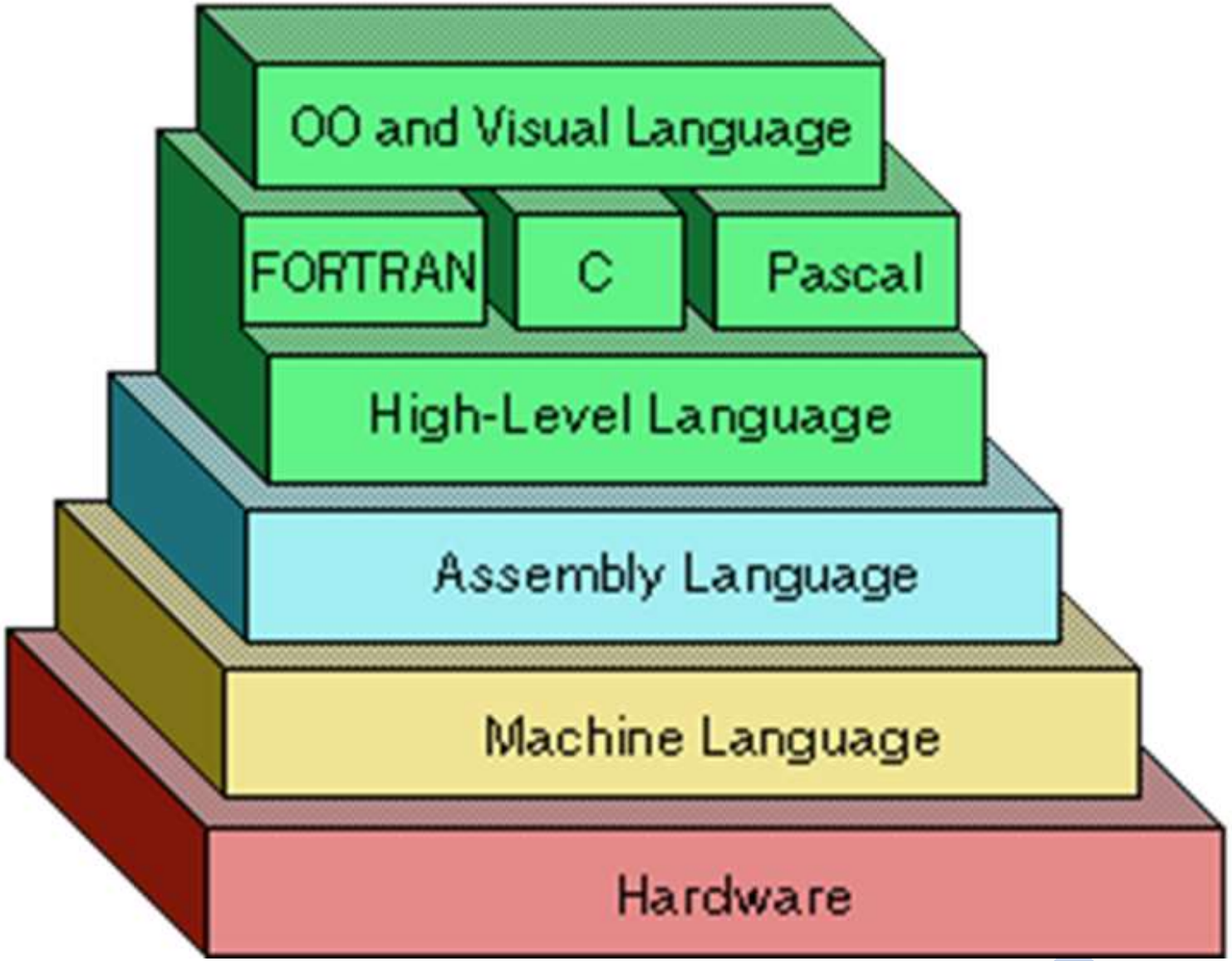
Block Diagram of 8086 Microprocessor

Hatırlatma

- Assembly dilinde yer alan tüm komutlar donanımsal sınırlamalar nedeniyle 8086 programlanırken kullanılmayabilir.

NEDİR?

Assembly programlama dili, kullanılan bilgisayar sisteminin yapısına ve işletim sistemi gibi platformlara sıkı-sıkıya bağımlı bir dildir. Assembly programlama dili düşük seviyeli bir dil olup C, C++, Pascal, C# gibi yüksek seviyeli programlama dillerine göre anlaşılması biraz daha zordur. Assembly dili ile program yazarken kullanılan bilgisayarın donanımsal özelliklerinin bilinmesi gerekir. Yazılan program kullanılan mikroişlemcinin yapısına bağlıdır. Assembly dili ile program yazarken programcı doğrudan bilgisayarın işlemcisi ve hafızası ile uğraşır. Anabellekteki (RAM'deki) ve işlemci kaydedicilerindeki değerleri doğrudan değiştirebilme imkanı vardır.

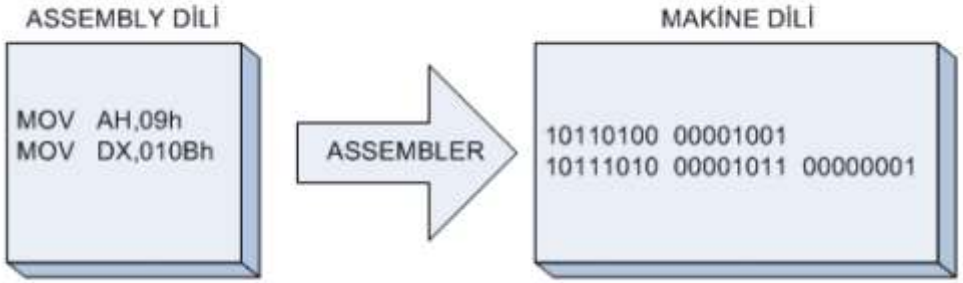

OO and Visual Language
FORTRAN
C
Pascal
High-Level Language
Assembly Language
Machine Language
Hardware

Mnemonic	Meaning	Condition	Mnemonic	Meaning	Condition
JA	above	CF = 0 and ZF = 0	JAE	above or equal	CF = 0
JB	below	CF = 1	JBE	below or equal	CF = 1 or ZF = 1
JC	carry	CF = 1	JCXZ	CX register is zero	(CF or ZF) = 0
JE	equal	ZF = 1	JG	greater	ZF = 0 and SF = OF
JGE	greater or equal	SF = OF	JL	less	(SF xor OF) = 1
JLE	less or equal	((SF xor OF) or ZF) = 1	JNA	not above	CF = 1 or ZF = 1
JNAE	not above nor equal	CF = 1	JNB	not below	CF = 0
JNBE	not below nor equal	CF = 0 and ZF = 0	JNC	not carry	CF = 0
JNE	not equal	ZF = 0	JNG	not greater	((SF xor OF) or ZF) = 1
JNGE	not greater nor equal	(SF xor OF) = 1	JNL	not less	SF = OF
JNLE	not less nor equal	ZF = 0 and SF = OF	JNO	not overflow	OF = 0
JNP	not parity	PF = 0	JNS	not sign	SF = 0
JNZ	not zero	ZF = 0	JO	overflow	OF = 1
JP	parity	PF = 1	JPE	parity even	PF = 1
JPO	parity odd	PF = 0	JS	sign	SF = 1
JZ	zero	ZF = 1			

	Addition
ADD	Add byte or word
ADC	Add byte or word with carry
INC	Increment byte or word by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
	Subtraction
SUB	Subtract byte or word
SBB	Subtract byte or word with borrow
DEC	Decrement byte or word by 1
NEG	Negate byte or word
AAS	ASCII adjust for subtraction
DAS	Decimal adjust for subtraction
	Multiplication
DIV	Divide byte or word unsigned
IDIV	Integer divide byte or word
AAD	ASCII adjust for division
CBW	Convert byte to word
CWD	Convert word to double word

Fig.14.4: Arithmetic instructions

ASSEMBLER NEDİR?



Mikroişlemci sadece ikili sayı sisteminde yazılan komut kodlarını, başka bir ifade ile makine dilinden anlar. Assembly dilinde yazılan programları makine diline çevirmek için Assembler adı verilen çevirici (derleyici) programlar kullanılır. Yanda verilen şekilde Assembly dili, Makine dili ve Assembler blok olarak görülmektedir.

NASIL ÇALIŞIR?

•Bilgisayarımızda çalıştırılan tüm programlar önce bilgisayarımızın RAM belleğine yüklenir. Daha sonra RAM bellekten sırası ile mikroişlemci tarafından okunarak çalıştırılır. RAM'e yüklenen veri programın makine dili karşılığından başka bir şey değildir. Yani 0 ve 1 kümeleridir.

•Makine dilinde program yazmak oldukça zordur. Buna karşılık makine dili ile birebir karşılığı olan ve komutları kısaltılmış kelimelerden (mnemonik) oluşan Assembly dilinden yararlanılır.

•Assembly dilinde program yazmak makine dilinde program yazmaya göre daha hızlı ve daha kolay yapılabilir. Ayrıca yazılan programların bellekte kapladıkları yerde aynıdır. Başka bir ifade ile bellek kullanımları aynıdır.

•Yüksek seviyeli dillerle karşılaştırıldığında assembly dilinde yazılan programlar daha hızlıdır ve bellekte daha az yer kaplar. Buna karşılık program yazmak yüksek seviyeli dillerde daha kolaydır.

•Assembly programlama dili günümüzde daha çok sistem programcıları tarafından diğer programlama dilleri içerisinde kullanılmaktadır.

DEZAVANTAJLARI

- Assembly dilinde program yazmak için mikroişlemci iç yapısı bilinmesi gerekir.
- Assembly dili mikroişlemci tipine göre değişir.
- Bir mikroişlemci için yazılan bir program başka bir mikroişlemcide çalışmayabilir. Program taşınabilir platformdan bağımsız değildir.
- Assembly dilinde program yazmak yüksek seviyeli dillere göre daha zor ve zaman alıcıdır.

C vs Assembly

```
main()
{
   int val1=10000h;
   int val2=40000h;
   int val3=20000h;
   int finalVal;

   finalVal = val1
      + val2 - val3;
}
```

```
.data
val1 DWORD 10000h
val2 DWORD 40000h
val3 DWORD 20000h
finalVal DWORD ?
.code
main PROC
   mov eax,val1
   add eax,val2
   sub eax,val3
   mov finalVal,eax
   call DumpRegs
   exit
main ENDP
```

AVANTAJLARI

•Bigisayar donanımı üzerinde daha iyi bir denetim sağlar. İşlemcinizin gücünü en iyi şekilde ortaya koyabilecek tek programlama dilidir.

•Küçük boyutlu bellekte az yer kaplayan programlar yazılabilir, virüslerin yazımında kullanılırlar.

•Yazılan programlar daha hızlı çalışır. Çok hızlı çalıştıkları için işletim sistemlerinde kernel ve donanım sürücülerinin programlanmasında, hız gerektiren kritik uygulamalarda kullanılmaktadır.

•Herhangi bir programlama dili altında, o dilin kodları arasında kullanılabilir.

•İyi öğrenildiğinde diğer dillerde karşılaşılan büyük problemlerin assembly ile basit çözümleri olduğu görülür.

NASIL YAZILIR?

- Assembly dilinde program yazmak için Windows altında yer alan notepad gibi herhangi bir metin editörü kullanılabilir.
- Metin editörü yardımı ile Assembly dilinde program yazılır. Yazılan program TASM veya MASM assembler çevirici programları yardımı ile .obj uzantılı olarak makine diline çevrilir.
- TASM = Turbo Assembler (a Borland product)
- MASM = Macro Assembler (a Microsoft product) – x86 opcode kullanırlar.
- Bu halde elde edilen program işletim sisteminin anladığı bir formatta değildir.
- TLINK bağlayıcı programı kullanılarak exe veya .com uzantılı hale dönüştürülür. Bu haldeki program işletim sistemi üzerinde ismi yazılarak DOS ortamında çalıştırılabilir.
- TLINK, kodun ihtiyaç duyduğu kütüphaneleri ve eklentileri bir exe paketi haline getirir.

Kaynaklar

- Feza Buzluca, İTÜ Ders Notları, Bilgisayar Mimarisi
- Wikipedia
- Emel Soylu, Kadriye Öz, Karabük Üniversitesi, Mikroişlemciler Ders Notları
- 1) Bilgisayar Mimarisi – Doç. Dr. Şirzat KAHRAMANLI
- 2) Ders Notları – Yrd. Doç. Dr. Rıfat KURBAN
- Wikipedia
- https://edukedar.com/difference-between-cisc-and-risc/
- Dr. B. B. Hegde First Grade College, Kundapura