BM-311 Bilgisayar Mimarisi

Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Konular

- Giriş
- Komut çalıştırma özellikleri
- Büyük register file kullanımı
- Compiler tabanlı register optimizasyonu
- RISC mimarisi
- RISC pipelining
- RISC ve CISC karşılaştırma

Giriş

- Bilgisayar organizasyonu ve mimarisi alanında bilgisayarın tarihsel gelişiminde önemli yenilikler yapılmıştır.
- **Family concept:** Farklı fiyat ve performansta aynı mimariye sahip bilgisayarlar üretilmiştir. İlk defa IBM tarafından 1964 yılında kullanılmıştır.
- **Mikroprogrammed control unit:** 1951 yılında Wilkes tarafından önerilmiştir. Kontrol biriminin tasarım ve gerçekleştirimini kolaylaştırmıştır.
- **Cache memory:** 1968 yılında ilk kez IBM tarafından kullanılmıştır. Günümüzde farklı seviyelerde, CPU içinde ve dışında kullanılmaktadır.

Giriş

- **Pipelining:** Farklı komutların farklı aşamaları eş zamanlı çalıştırılır.
- Performans, oluşturulan aşama sayısına bağlı olarak artmıştır.
- **Multiple processors:** Çok sayıda farklı organizasyonu içerir.
- Birden fazla işlemci aynı işin farklı kısımlarını gerçekleştirebilir.
- Hafıza paylaşımı yapılabilir ve her CPU için farklı cache bellekler kullanılabilir.
- **RISC:** CISC mimarisinin alternatifi olarak geliştirilmiştir. Temel özellikleri:
  - Çok sayıda genel amaçlı register
  - Register kullanımını optimize eden compiler
  - Az sayıda ve basit komutlar
  - Daha iyi pipelining performansı

Giriş

- RISC ve RISC olmayan makinelerin karşılaştırması.

	Complex Instruction Set (CISC) Computer			Reduced Instruction Set (RISC) Computer		Superscalar		
Characteristic	IBM 370/168	VAX 11/780	Intel 80486	SPARC	MIPS R4000	PowerPC	Ultra SPARC	MIPS R10000
Year developed	1973	1978	1989	1987	1991	1993	1996	1996
Number of instructions	208	303	235	69	94	225		
Instruction size (bytes)	2–6	2–57	1–11	4	4	4	4	4
Addressing modes	4	22	11	1	1	2	1	1
Number of general-purpose registers	16	16	8	40 - 520	32	32	40 - 520	32
Control memory size (Kbits)	420	480	246	—	—	—	—	—
Cache size (KBytes)	64	64	8	32	128	16-32	32	64

Konular

- Giriş
- Komut çalıştırma özellikleri
- Büyük register file kullanımı
- Compiler tabanlı register optimizasyonu
- RISC mimarisi
- RISC pipelining
- RISC ve CISC karşılaştırma

Komut çalıştırma özellikleri

- Yüksek seviyeli dillerdeki (HLL-High Level Language) gelişmeler donanımdaki gelişmelerden çok daha hızlı gerçekleşmiştir.
- Yüksek seviyeli dillerle bilgisayar mimarisi arasındaki fark **(semantic gap) giderek artmıştır.**
- Bilgisayar mimarisinde bu farkı azaltmak için **daha karmaşık ve daha çok sayıda komut oluşturulmuştur.**
- Komutlardaki **adresleme modlarının sayısı artırılmıştır.**
- **Karmaşık komut kümeleri geliştirmenin amaçları:**
  - Compiler yazımını kolaylaştırmak,
  - Mikrokod seviyesinde karmaşık komutlar geliştirerek çalışma performansını artırmak,
  - Yüksek seviyeli dilleri daha çok destekleyen komutlar geliştirmek.

Komut çalıştırma özellikleri

- Karmaşık komut kümelerine sahip bilgisayarların (CISC) performansını artırmak için çok sayıda teknik geliştirilmiştir.
- Yüksek seviyeli dillerle yazılan **programların CISC bilgisayarlarda çalışması analiz edilmiştir:**
  - **Operations performed**
    CPU tarafından gerçekleştirilen işlemler ve hafıza erişimleri incelenmiştir.
  - **Operands used**
    Operandların kullanım frekansları ve adresleme modları incelenmiştir.
  - **Execution sequencing**
    Kontrol ve pipeline organizasyonu incelenmiştir.
- **Yapılan çalışmalarda** programlar çalışırken **dinamik sonuçlar alınmıştır.**
- Programların tümü yüksek seviyeli dillerle yazılmıştır.

Komut çalıştırma özellikleri

Operations

- **Sonuçlar dinamik çalışmayla alınmıştır.**

	Dynamic Occurrence		Machine-Instruction Weighted		Memory-Reference Weighted	
	Pascal	C	Pascal	C	Pascal	C
ASSIGN	45%	38%	13%	13%	14%	15%
LOOP	5%	3%	42%	32%	33%	26%
CALL	15%	12%	31%	33%	44%	45%
IF	29%	43%	11%	21%	7%	13%
GOTO	—	3%	—	—	—	—
OTHER	6%	1%	3%	1%	2%	1%

- **Dynamic occurence,** programların dinamik olarak çalışması sırasında komutların çalışma sıklıklarıdır.
- **Machine-instruction weighted,** komutların çalışma sıklıkları ile compiler tarafından kendisi için oluşturulan makine komutu sayısının çarpımıdır.
- **Memory-reference weighted,** komutların çalışma sıklıkları ile her komutun hafızaya erişim sıklığının çarpımıdır.

Komut çalıştırma özellikleri

Operations - devam

- **Atama deyimleri (assignments) ağırlıklı olarak çalıştırılmaktadır.**
  - Data movement instructions
- **Şartlı atlama deyimleri sıklıkla kullanılmaktadır.**
  - if, loop, program akış kontrolü
- **Prosedür çağırma ve geri dönme komutları çok zaman harcamaktadır.**
  - call, return
- **Bazı HLL komutları çok sayıda makine komutu ile oluşturulmaktadır.**

Komut çalıştırma özellikleri

Operands

- **Programlarda** çalışma sırasında **çoğunlukla lokal skalar değişkenler kullanılmaktadır.**
- Lokal değişkenlere erişim optimizasyonu yapılmalıdır.

	Pascal	C	Average
Integer constant	16%	23%	20%
Scalar variable	58%	53%	55%
Array/Structure	26%	24%	25%

Komut çalıştırma özellikleri

Procedure call/return

- Procedure çağırmalarda parametre sayısı ve iç içe çağırma sayısı önemlidir.
- Procedure'lerden **%98'i 6'dan daha az parametre almaktadır.**
- Procedure'lerin **%92'si 6'dan daha az lokal değişken kullanmaktadır.**

Komut çalıştırma özellikleri

Sonuçların değerlendirmesi

- Yüksek seviyeli dilleri desteklemek için **geliştirilen komut kümeleri çok etkili bir yöntem olmamıştır.**
- Bunun yerine **çok zaman harcayan işlemlerin optimizasyonu daha uygundur.**
- Operandların saklanması için **çok sayıda register oluşturulmalıdır.**
- **Şartlı atlama** ve **procedure çağırma** komutlarındaki **pipeline performansının artırılması gereklidir.**
- Basit komut kümesine sahip bilgisayar (Reduced Instruction Set Computer-RISC) oluşturulmalıdır.

Konular

- Giriş
- Komut çalıştırma özellikleri
- Büyük register file kullanımı
- Compiler tabanlı register optimizasyonu
- RISC mimarisi
- RISC pipelining
- RISC ve CISC karşılaştırma

Büyük register file kullanımı

- Çok **sık erişilen operand'lar register'larda saklanmalıdır.**
- **Yazılım** veya **donanım** yaklaşımıyla **register kullanımı maksimize edilmeye çalışılmaktadır.**
- **Donanım yaklaşımında, register sayısı artırılmaktadır.**
- Daha fazla operand daha uzun süre register'larda tutulmakta ve performans artırılmaktadır.
- **Yazılım yaklaşımında, compiler tarafından register kullanımı maksimum yapılır.**
- Böylece **hafıza erişimi minimum yapılmaya çalışılır.**
- Procedure **parametre gönderme** ve **sonuç alma** işlemleri de **register'lar üzerinde yapılır.**

Büyük register file kullanımı

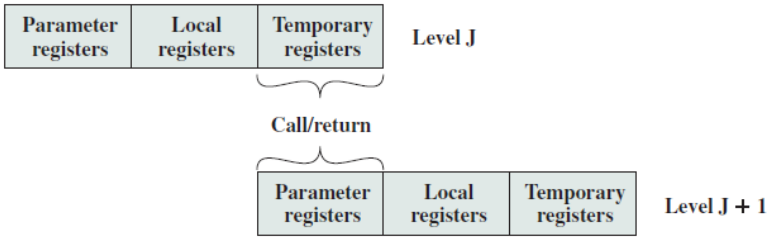
Register windows

- Procedure **çağırma** ve **geri dönme** işlemlerinde hafıza yerine **register kullanılmalıdır.**
- Çok sayıda küçük register kümesi kullanılır.
- Her **call işleminde** yeni bir **register kümesine switch yapılır.**
- Her **return işleminde** bir **önceki register kümesine switch yapılır.**
- **Call/return işlemlerinde** gönderilen ve alınan **parametreler aynı register grubunda üst üste gelir.**

Büyük register file kullanımı

Register windows – devam

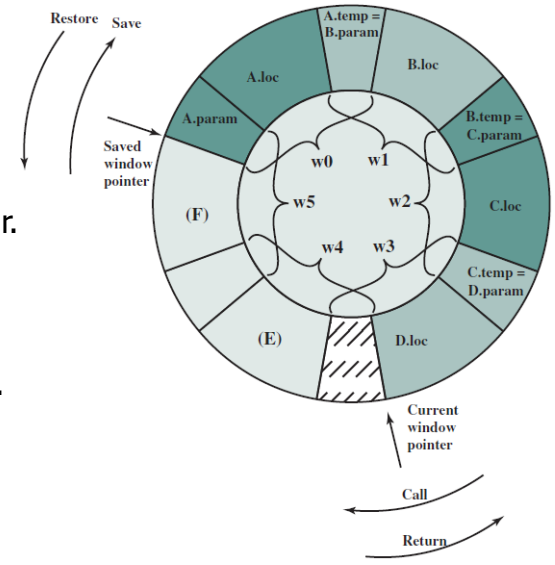
- **Parameter registers** kısmına gönderilen ve alınan operandlar yerleştirilir.
- **Local registers** kısmında procedure içindeki lokal değişkenler tutulur.



Büyük register file kullanımı

Register windows – devam

- Register windows yönteminde en son geçilen procedure işlemleri saklanır, öncekiler hafızada tutulur.
- Bunun için dairesel buffer yapısı **(circular buffer organization)** kullanılır.



Büyük register file kullanımı

Register windows – devam

- Her **yeni call işleminde saat yönünde ilerlenir.**
- Her **return işleminde saat yönünün tersine ilerlenir.**
- Tüm pencereler kullanıldığında bir interrupt üretilerek eski pencere hafızaya aktarılır.
- Return işleminde ise hafızaya aktarılana dönünceye kadar hafızadan geri alma işlemi yapılmaz.

Büyük register file kullanımı

Global değişkenler

- **Compiler tarafından tüm global değişkenler hafızaya yerleştirilir.**
- Sık erişilenler için etkili yöntem değildir.
- Alternatif olarak, **CPU içinde global register kümesi oluşturulur ve global değişkenler burada tutulur.**
- Register optimizasyonu compiler tarafından yapılır.

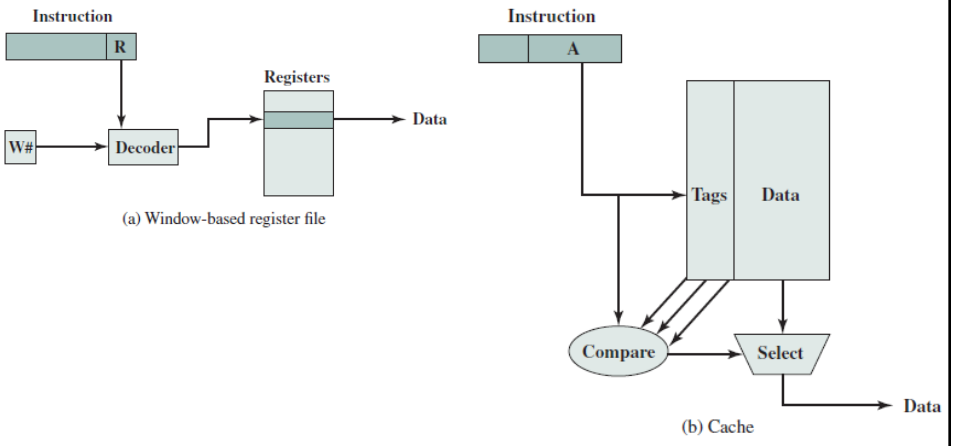
Büyük register file kullanımı

Büyük register file ile önbellek karşılaştırması

Large Register File	Cache
All local scalars	Recently-used local scalars
Individual variables	Blocks of memory
Compiler-assigned global variables	Recently-used global variables
Save/Restore based on procedure nesting depth	Save/Restore based on cache replacement algorithm
Register addressing	Memory addressing
Multiple operands addressed and accessed in one cycle	One operand addressed and accessed per cycle

Büyük register file kullanımı

Skalar değişkenin seçilmesi



Konular

- Giriş
- Komut çalıştırma özellikleri
- Büyük register file kullanımı
- Compiler tabanlı register optimizasyonu
- RISC mimarisi
- RISC pipelining
- RISC ve CISC karşılaştırma

Compiler tabanlı register optimizasyonu

- Register optimizasyonu compiler tarafından yapılır.
- Compiler **operand'ları olabildiğince register'larda tutmaya çalışır.**
- **Load/store** komutlarının **kullanım sayısını azaltmak amaçlanmıştır.**
- **Her değişkene sembolik register atanır** daha sonra **gerçek register'larla eşleştirilir.**
- Sembolik register'lardan **eş zamanlı kullanılmayanlar aynı gerçek register'ı paylaşır.**
- Eş zamanlı kullanılan register sayısı gerçek register sayısından fazla ise bazı operand'lar hafızaya atanır.

Compiler tabanlı register optimizasyonu

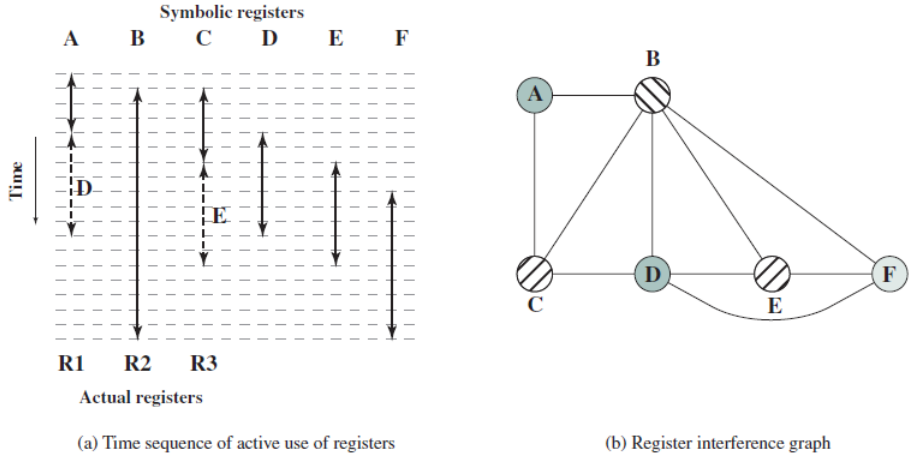
Graph coloring

- **Her node bir sembolik register'ı gösterir.**
- Register kullanımı için zaman akışı belirlenir.
- **Komşu node'lar eş zamanlı kullanılan register'ları gösterir.**
- Komşu node'ların farklı renklerde boyanması gerekir.
- Kullanılabilecek maksimum **renk sayısı gerçek register sayısına eşittir.**
- Her renk bir gerçek register'ı gösterir.

Compiler tabanlı register optimizasyonu

Graph coloring – devam

- Şekilde F boyanamamıştır ve load/store yapılacaktır.



(a) Time sequence of active use of registers

(b) Register interference graph

Konular

- Giriş
- Komut çalıştırma özellikleri
- Büyük register file kullanımı
- Compiler tabanlı register optimizasyonu
- RISC mimarisi
- RISC pipelining
- RISC ve CISC karşılaştırma

RISC mimarisi

CISC mimarisinin kullanımı

- Daha basit compiler geliştirilir.
  - **Karmaşık makine komutlarının çalıştırılması daha zordur.**
  - Pipeline optimizasyonu daha zordur.
- Daha küçük programlar geliştirilir.
  - **Programlar hafızada daha az yer kaplar.**
  - Hafıza fiyatları ilk yıllara göre düşmüştür.
  - Çoğu makine komutu uzun opcode'a sahiptir, ancak register adresleme daha az bit gerektirir.
- Programların çalışma hızı düşüktür.
  - **Kontrol birimi daha karmaşıktır.**
  - **Mikroprogramlanmış kontrol birimi daha yavaş çalışır.**
  - Basit komutların çalışması uzun zaman alabilir.

RISC mimarisi

RISC mimarisinin karakteristik özellikleri

- **Her cycle'da bir komut çalıştırma** (Bir cycle iki reg operand ile aritmetik işlem ve reg saklama süresi, ADD AX, BX, CX)
- **Register-register** işlemleri (RISC mimarisinde bir veya iki tane ADD komutu, CISC mimarisinde 20-25 ADD komutu)
- **Basit ve az sayıda adresleme modu**
- **Basit komut formatları**
- **Hardwired kontrol birimi** (mikroprogramlanmış kontrol birimi kullanılmaz.)
- Komutlar için **mikrokod kullanılmaz.**
- **Sabit komut formatı**
- Daha **karmaşık compiler**
- **Interrupt tepki süresi RISC mimarisinde daha iyidir.**

RISC mimarisi

CISC ve RISC mimarisinin karşılaştırılması

1. RISC **sabit komut uzunluğuna sahiptir** (4 byte).
2. RISC **az sayıda adresleme moduna sahiptir** (5'ten az).
3. RISC **indirect adresleme modu kullanmaz.**
4. RISC **aritmetik işlemlerde load/store yapmaz.**
5. RISC komutlarda **birden fazla hafıza adresleme kullanmaz.**
6. RISC **integer register adresleme için 5 veya daha fazla bit kullanır** (en az 32 integer register).
7. RISC **floating-point register adresleme için 4 veya daha fazla bit kullanır** (en az 16 floating-point register).

- **1-2 decode işlem karmaşıklığını, 3-5 pipeline performansını, 6-7 compiler performansını belirler.**

Konular

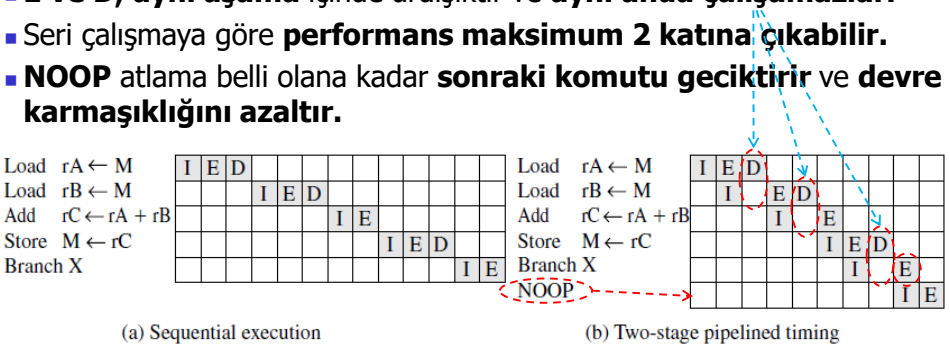
- Giriş
- Komut çalıştırma özellikleri
- Büyük register file kullanımı
- Compiler tabanlı register optimizasyonu
- RISC mimarisi
- RISC pipelining
- RISC ve CISC karşılaştırma

RISC pipelining

- Komutların çoğu **register-register adresleme** yapar.
- **Load/store dışındaki tüm komutlar** 2 alt işlemden oluşur:
  - **I:** Instruction fetch
  - **E:** Execute **(Decode+ALU işlemi+giriş ve çıkışlar register)**
- **Load/store komutları** 3 alt işlemden oluşur:
  - **I:** Instruction fetch
  - **E:** Execute **(Decode+ALU ile hafıza adresi hesaplanır.)**
  - **D:** Memory **(Register-to-memory, memory-to-register işlemi yapılır.)**

RISC pipelining

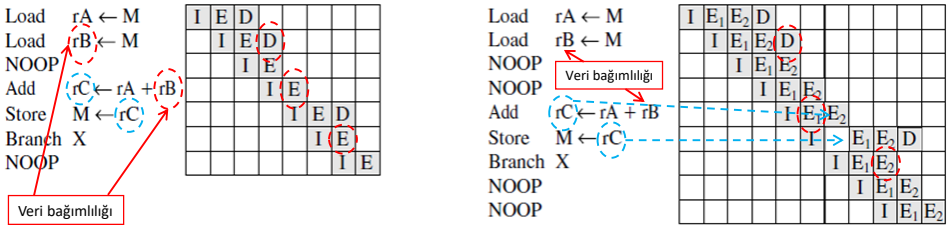
- **Sıralı çalışma, 2 aşamalı, 3 aşamalı** ve **4 aşamalı pipeline ile çalışma** aşağıdaki gibi gerçekleşir.
- Sıralı çalışmada performans çok düşüktür.
- **İki aşamalı** pipeline ile **I (Fetch)** ile **E+D (Execute+Memory)** aşamaları **eş zamanlı çalışabilir.**
- **E ve D, aynı aşama** içinde ardışıktır ve **aynı anda çalışamazlar.**
- Seri çalışmaya göre **performans maksimum 2 katına çıkabilir.**
- **NOOP** atlama belli olana kadar **sonraki komutu geciktirir** ve **devre karmaşıklığını azaltır.**



(a) Sequential execution (b) Two-stage pipelined timing

RISC pipelining

- İki aşamalı pipeline'daki **E (Execute) ve D (Memory) işlemleri ayrıştırılıp üç aşamalı pipeline oluşturulabilir.**
- Üç farklı komut aynı anda işleme alınabilir.
- Seri çalışmaya göre **performans maksimum 3 katına çıkabilir.**
- **Veri bağımlılığı ve atlama komutları performansı düşürür.**
- E aşaması **decode+register'dan okuma (E_1)** ile **ALU işlemi ve register'a yazma (E_2)** olarak ayrıştırılabilir **(4 aşama)**.
- Seri çalışmaya göre **performans maksimum 4 kata çıkabilir.**



(c) Three-stage pipelined timing (d) Four-stage pipelined timing

RISC pipelining

Pipeline optimizasyonu

- Ardışık komutlardaki data ve branch bağımlılığı pipeline performansını düşürür.
- **Delayed branch** ile **komutların çalışma sırası yeniden düzenlenir.**
- Delayed branch ile **branch komutunun sonucu belli olana kadar bağımsız başka komutlar çalıştırılır.**

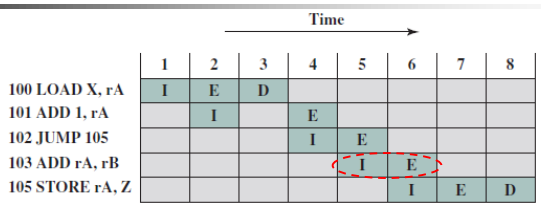
RISC pipelining

Delayed branch

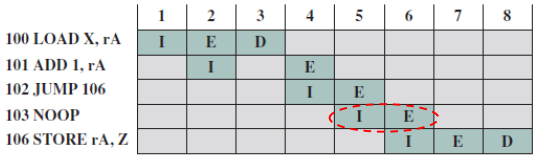
Address	Normal branch	Delayed branch	Optimized delayed branch
100	LOAD X, rA	LOAD X, rA	LOAD X, rA
101	ADD 1, rA	ADD 1, rA	**JUMP 105**
102	**JUMP 105**	JUMP 106	**ADD 1, rA**
103	ADD rA, rB	**NOOP**	ADD rA, rB
104	SUB rC, rB	ADD rA, rB	SUB rC, rB
105	**STORE rA, Z**	SUB rC, rB	**STORE rA, Z**
106		**STORE rA, Z**	

RISC pipelining

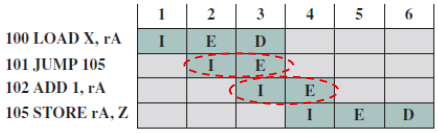
Delayed branch



(a) Traditional pipeline



(b) RISC pipeline with inserted NOOP



(c) Reversed instructions

Konular

- Giriş
- Komut çalıştırma özellikleri
- Büyük register file kullanımı
- Compiler tabanlı register optimizasyonu
- RISC mimarisi
- RISC pipelining
- RISC ve CISC karşılaştırma

RISC ve CISC karşılaştırma

- Uzun yıllar **bilgisayar mimarisinde ve organizasyonunda aşağıdaki geliştirmeler yapılmıştır:**
  - İşlemci karmaşıklığı
  - Daha çok komut
  - Daha çok adresleme modu
  - Daha çok özel amaçlı register
- **RISC mimarisi** bu gelişmeleri tümüyle tersine çevirmiştir ve **daha basit işlemci tasarımını ön plana çıkarmıştır.**
- **Günümüzde RISC ve CISC mimarileri birbirinden bazı özelliklerini alarak geliştirilmeye devam etmektedir.**
- **RISC** mimarisi giderek **karmaşıklaşmakta, CISC** mimarisi ise **daha çok register kullanmaktadır.**

RISC ve CISC karşılaştırma

- RISC ve CISC mimarilerini karşılaştırmak için çok sayıda çalışma yapılmıştır.
- Yapılan karşılaştırmalarda aşağıdaki sorunlar vardır:
  - Birbirine **denk RISC ve CISC makine olmadığı için** yaşam süresindeki **kullanım maliyeti, teknoloji seviyesi, devre karmaşıklığı, compiler karmaşıklığı** ve **işletim sistemi desteği** konularında **yeterince karşılaştırma yapılamamaktadır.**
  - Kabul edilen bir test programı yoktur, **performans kullanılan programlara göre değişebilmektedir.**
  - Ticari olarak üretilen **RISC makineler CISC özelliklerine de sahiptir** ve ticari CISC makineyle karşılaştırılması zordur.
  - Teknolojideki gelişmeler sonucunda **RISC makineler de giderek karmaşık hale gelmiştir.**