BM-311 Bilgisayar Mimarisi

Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Konular

- Giriş
- Superscalar işlemciler
- Superscalar ve superpipeline
- Paralel çalışmadaki problemler
- Tasarım özellikleri
- Komut çalıştırma kuralları
- Register renaming
- Makine paralelliği
- Branch prediction
- Superscalar çalışma

Giriş

- Superscalar mimaride **integer** ve **floating-point aritmetik komutlar, şartlı atlama komutları** birbirinden bağımsız ve **eşzamanlı çalıştırılır.**
- Superscalar yapı RISC ve CISC mimarisinde kullanılmaktadır.
- RISC mimarisinde kullanımı daha yaygındır.
- Superscalar yapı **komut seviyesinde paralel çalışmayı destekler.**
- Superscalar işlemci **birden fazla komutu eşzamanlı fetch eder** ve **birbirinden bağımsız olanları eş zamanlı çalıştırır.**
- Birbirine bağımlı olanların bağımlılıklarını ortadan kaldırmak için farklı yöntemler uygulanır.

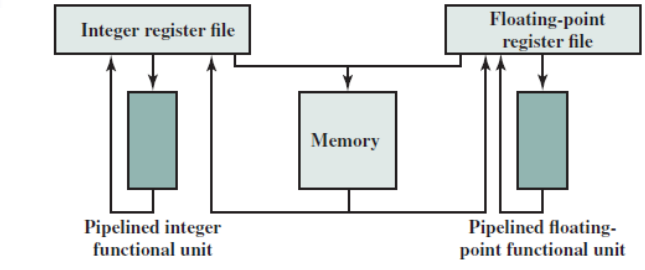
Konular

- Giriş
- Superscalar işlemciler
- Superscalar ve superpipeline
- Paralel çalışmadaki problemler
- Tasarım özellikleri
- Komut çalıştırma kuralları
- Register renaming
- Makine paralelliği
- Branch prediction
- Superscalar çalışma

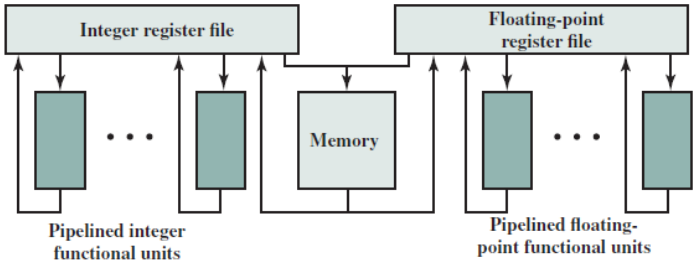
Superscalar işlemciler

- Superscalar ilk defa 1987 yılında kullanılmaya başlanmıştır.
- Superscalar yaklaşımın temeli, birbirinden bağımsız **komutların ayrı pipeline'larda eş zamanlı** çalıştırılmasına dayanır.
- Superscalar işlemcilerde **fonksiyonel birimlerden birden fazla bulunur.**
- Birden fazla aritmetik işlem (tamsayı toplama/çarpma, floating point toplama/çarpma) aynı anda yapılabilir.

Superscalar işlemciler



(a) Scalar organization



(b) Superscalar organization

Superscalar işlemciler

- Yapılan çalışmalarda elde edilen performans artışı aşağıdaki tablodaki gibidir.

Kaynaklar*	Performans artışı
[TJAD70]	1.8
[KUCK72]	8
[WEIS84]	1.58
[ACOS86]	2.7
[SOHI90]	1.8
[SMIT89]	2.3
[JOUP89b]	2.2
[LEE91]	7

- Performans değerlerindeki farklılıklar, simülasyon yapılan donanım ve yazılımların farklı olmasından kaynaklanmaktadır.

*Kaynaklar son sayfada verilmiştir.

Konular

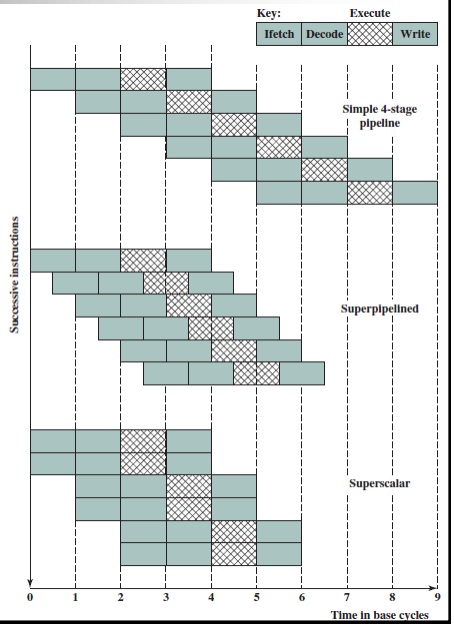
- Giriş
- Superscalar işlemciler
- Superscalar ve superpipeline
- Paralel çalışmadaki problemler
- Tasarım özellikleri
- Komut çalıştırma kuralları
- Register renaming
- Makine paralelliği
- Branch prediction
- Superscalar çalışma

Superscalar ve superpipeline

- Performansı artırmak için uygulanan diğer bir yaklaşım superpipeline'dır.
- Superpipeline ilk defa 1988 yılında uygulanmaya başlanmıştır.
- Superpipeline, **birçok pipeline aşamasının bir clock cycle'ın yarısından kısa sürede bitmesinden dolayı clock cycle hızını iki katına (2.derece) çıkartır.**
- MIPS R4000 superpipeline yaklaşımını kullanmaktadır.

Superscalar ve superpipeline

- **4 aşamalı pipeline**
- **2.derece superpipeline**
- **Superscalar**



Konular

- Giriş
- Superscalar işlemciler
- Superscalar ve superpipeline
- Paralel çalışmadaki problemler
- Tasarım özellikleri
- Komut çalıştırma kuralları
- Register renaming
- Makine paralelliği
- Branch prediction
- Superscalar çalışma

Paralel çalışmadaki problemler

- Superscalar yaklaşımda komutların paralel çalıştırılması amaçlanır.
- **Instruction-level paralel çalışma**, program komutlarının paralel çalıştırılmasını ifade eder.
- Paralel çalışma optimizasyonu **compiler** veya **donanım tarafından yapılabilir.**
- Paralel çalışma sırasında **aşağıdaki problemler** oluşabilir:
  - True data dependency (write-read dependency)
  - Procedural dependency
  - Resource conflicts
  - Output dependency (write-write dependency)
  - Antidependency (read-write dependency)

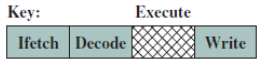
Paralel çalışmadaki problemler

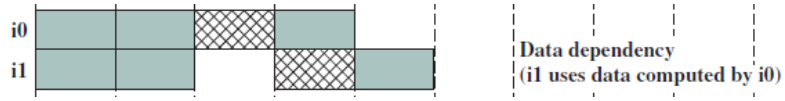
True data dependency (write-read dependency)

- Aşağıdaki kod parçasında true data dependency (flow dependency, **read after write (RAW)** dependency) vardır.

```
ADD EAX, ECX ;load register EAX with the con-
             ;tents of ECX plus the contents
             ;of EAX
MOV EBX, EAX ;load EBX with the contents of EAX
```

- İkinci komut fetch ve decode edilir, ancak birinci komut execute yapılıncaya kadar execute edilemez.

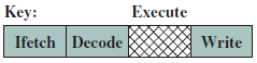


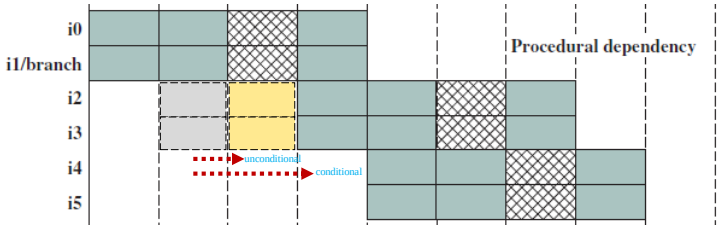


Paralel çalışmadaki problemler

Procedural dependency

- Bir **branch komutundan sonraki komut(lar) atlama işleminin sonucuna bağlıdır** (procedural dependency).
- Branch execute edilmeden çalıştırılamaz.

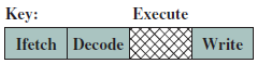


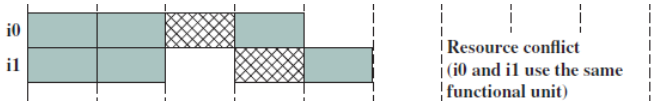


Paralel çalışmadaki problemler

Resource conflicts

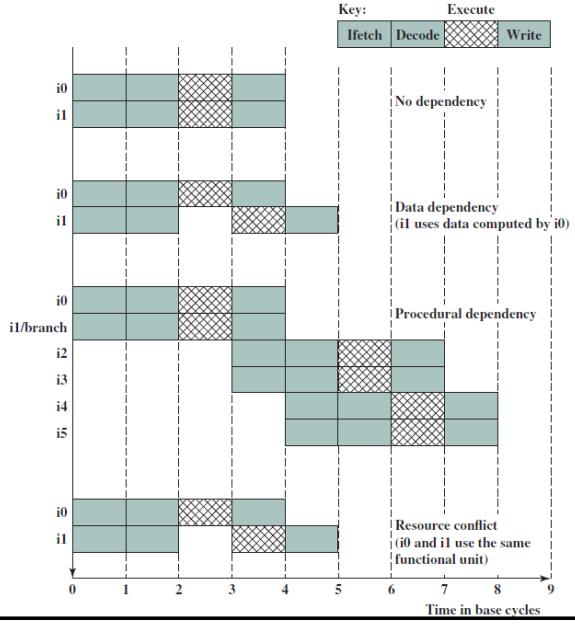
- Kaynak çakışması **birden fazla komutun aynı işlem birimini kullanmasından kaynaklanır.**
- Kaynaklar hafıza, bus, cache, register-file port ve fonksiyonel birimlerdir (ALU toplayıcı vb.).
- Resource conflict, **birden fazla fonksiyonel birim oluşturularak giderilebilir.**





Paralel çalışmadaki problemler

- No dependency
- Data dependency
- Procedural dependency
- Resource conflict



Konular

- Giriş
- Superscalar işlemciler
- Superscalar ve superpipeline
- Paralel çalışmadaki problemler
- Tasarım özellikleri
- Komut çalıştırma kuralları
- Register renaming
- Makine paralelliği
- Branch prediction
- Superscalar çalışma

Tasarım özellikleri

Instruction-level parallelism ve machine parallelism

- **Komut seviyesinde paralellik** birbirine bağımlı olmayan **komutların eş zamanlı çalışmasını ifade eder.**
- **Soldaki** kod parçasında **üç komut bağımsızdır** ve eş zamanlı çalışır.
- **Sağdaki** kod parçasında **üç komut birbirine bağımlıdır** ve eş zamanlı çalıştırılamazlar.

```
Load R1, R2          Add R3, "1"
Add  R3, "1"         Add R4, R3
Add  R4, R2          Store [R4], R0
```

- **Machine parallelism**, birden çok komutun fetch ve execute edilmesi için **birden fazla pipeline kullanır.**

Konular

- Giriş
- Superscalar işlemciler
- Superscalar ve superpipeline
- Paralel çalışmadaki problemler
- Tasarım özellikleri
- Komut çalıştırma kuralları
- Register renaming
- Makine paralelliği
- Branch prediction
- Superscalar çalışma

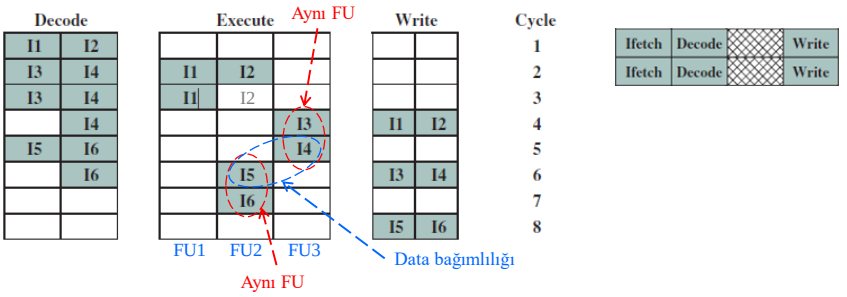
Komut çalıştırma kuralları

- Pipeline aşamalarının birden fazla kullanılması halinde **her aşamada birden çok komutun yönetilmesi gerekir.**
- **Instruction issue,** bir komutun **decode aşamasından execute aşamasına geçmesini** (fonksiyonel birimde çalıştırılmaya başlatılması) ifade eder.
- Komutların **fetch (FI) sırası, execute (EI) sırası** ve **register** veya **hafıza içeriğini değiştirme sırası (WO)** önemlidir.
- Komut çalıştırma kuralları:
  - **In-order-issue with in-order-completion** (sıralı alma ve sıralı tamamlama)
  - **In-order-issue with out-of-order-completion** (sıralı alma ve sırasız tamamlama)
  - **Out-of-order-issue with out-of-order-completion** (sırasız alma ve sırasız tamamlama)

Komut çalıştırma kuralları

In-order-issue with in-order-completion

- Komutlar **sıralı execute edilirler** ve **aynı sırada sonuçlarını** yazarlar.
- Pipeline, **2 komutu** aynı anda **fetch ve decode** ediyor, **3 fonksiyonel birim** var, **2 write-back** aşamasına sahip.

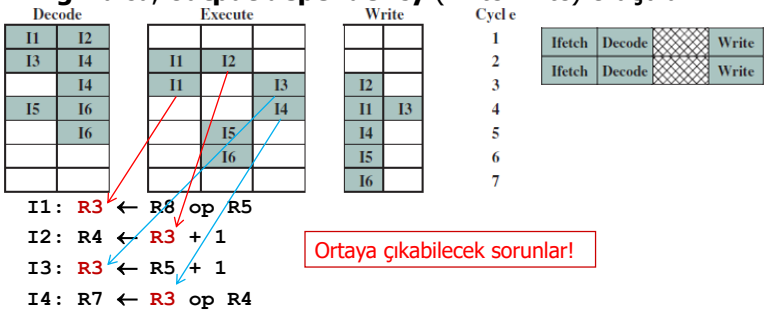


- **I1 execute** için **2 cycle** gerektiriyor. **I1** ve **I2** arasında bağımlılık yok.
- **I3 ve I4** aynı fonksiyonel birimi kullanıyor **(resource conflict).**
- **I5** komutu **I4** komutunun sonucuna bağımlıdır **(data dependency)**.
- **I5 ve I6** aynı fonksiyonel birimi kullanıyor **(resource conflict).**

Komut çalıştırma kuralları

In-order-issue with out-of-order-completion

- **I2** komutu **I1** komutunun execute edilmesini **beklemeden execute edilir.**
- **I3 komutu daha erken tamamlanır** (tümü 1 cycle önce biter).
- **Data bağımlılığı** varsa, **output dependency** (write-write) **oluşabilir.**

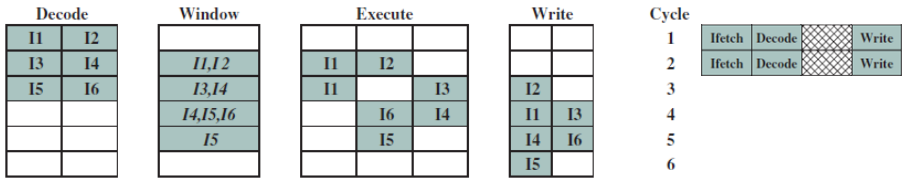


- **I2** komutu **I1** execute edilmeden execute edilemez (true data dependency).
- **I4** komutu **I3** komutunu beklemek zorundadır (true data dependency).
- **I3** komutu **I1** komutu execute edilmeden execute edilemez (output dependency). Aksi halde **I4 komutunun sonucu yanlış olur.**

Komut çalıştırma kuralları

Out-of-order-issue with out-of-order-completion

- **In-order-issue** yöntemlerinde **resource conflict veya dependency varsa, yeni bir komut decode edilememektedir.**
- **Out-of-order issue** yönteminde **instruction window** kullanılarak decode edilen komutlar burada **bekletilir**.



- Window içindeki komutlardan **beklediği kaynak boşalan execute edilir.**
- Komutun diğer komutlarla kaynak veya data bağımlılığı olmamalıdır.
- Örnekte, **I5 ile I4 arasında bağımlılık vardır (data dependency).**
- **I6 ile I4 arasında bağımlılık yoktur.**

Komut çalıştırma kuralları

Antidependency (read-write dependency)

- **Out-of-order-issue with out-of-order-completion** yönteminde **write-read, read-write, write-write** dependency oluşabilir.

```
I1: R3 ← R8 op R5
I2: R4 ← R3 + 1
I3: R3 ← R5 + 1
I4: R7 ← R3 op R4
```

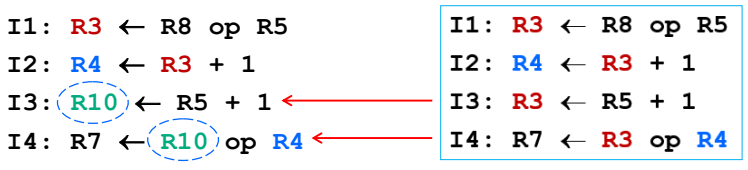
- **I2** ile **I3** arasında **antidependency** (read-write) vardır.
- **I2** komutu execute'a başlamadan önce **I3** komutu execute edilemez. Aksi durumda, **I2 komutunun sonucu yanlış olur.**

Konular

- Giriş
- Superscalar işlemciler
- Superscalar ve superpipeline
- Paralel çalışmadaki problemler
- Tasarım özellikleri
- Komut çalıştırma kuralları
- Register renaming
- Makine paralelliği
- Branch prediction
- Superscalar çalışma

Register renaming

- **Read-write (RAW)** veya **write-write (WAW) dependency** oluştuğunda pipeline önceki komutu bekler.
- Register renaming ile **register'lar dinamik olarak atanır.**
- Renaming yapılan register **programın devam eden kısmında komple değiştirilir.**



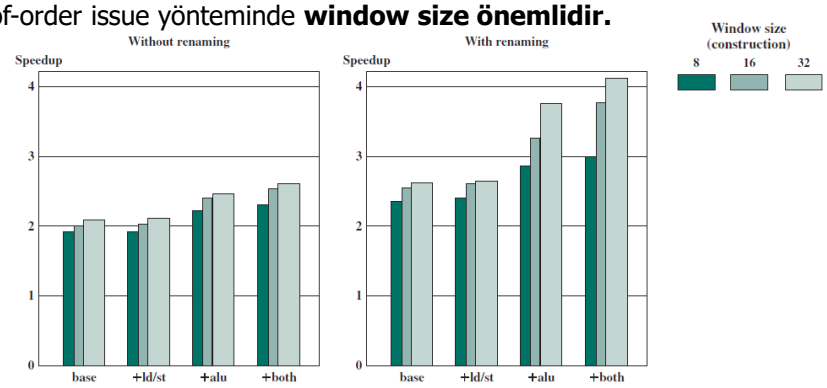
- **I3** komutu ile **I2** komutu arasındaki antidependency **(read-write)** (R3->R10) giderilmiştir.
- **I3** komutu ile **I1** komutu arasındaki output dependency **(write-write)** (R3->R10) giderilmiştir.

Konular

- Giriş
- Superscalar işlemciler
- Superscalar ve superpipeline
- Paralel çalışmadaki problemler
- Tasarım özellikleri
- Komut çalıştırma kuralları
- Register renaming
- Makine paralelliği
- Branch prediction
- Superscalar çalışma

Makine paralelliği

- **Kaynak sayıları artırılarak**, out-of-order issue ve register renaming ile paralel çalışmadaki **performans artırılabilir.**
- Out-of-order issue yönteminde **window size önemlidir.**



- Base makinede kaynaklar tektir, ancak out-of-order issue yapılmaktadır.
- Sırasıyla **ld/st birimi**, **ALU** ve **ld/st ile ALU** birimlerinin ikisinin de **iki adet olduğu** durumlarda window boyutu arttıkça performans artmaktadır.
- **Register renaming** ile **performans artışı daha yüksektir.**

Konular

- Giriş
- Superscalar işlemciler
- Superscalar ve superpipeline
- Paralel çalışmadaki problemler
- Tasarım özellikleri
- Komut çalıştırma kuralları
- Register renaming
- Makine paralelliği
- Branch prediction
- Superscalar çalışma

Branch prediction

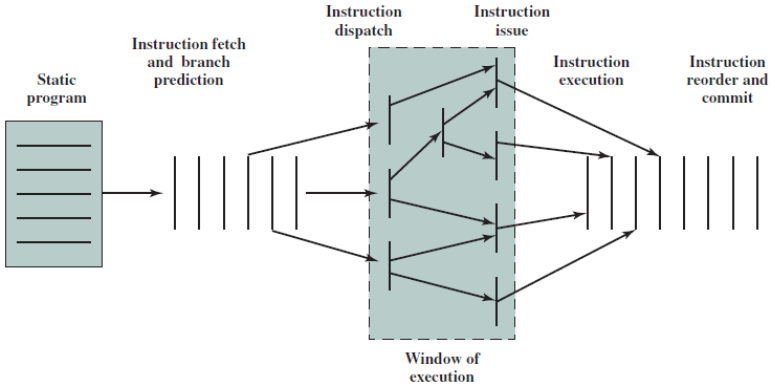
- Intel 80486 işlemci, **hem sonraki hem de target adresteki komutu fetch eder** (speculatively fetch).
- **RISC** makineler **delayed branch** yöntemini kullanır.
- **Superscalar** işlemciler klasik **branch prediction** yöntemlerini kullanır.

Konular

- Giriş
- Superscalar işlemciler
- Superscalar ve superpipeline
- Paralel çalışmadaki problemler
- Tasarım özellikleri
- Komut çalıştırma kuralları
- Register renaming
- Makine paralelliği
- Branch prediction
- Superscalar çalışma

Superscalar çalışma

- Superscalar işlemciler komutları **branch prediction yöntemlerini de kullanarak fetch eder.**
- **Instruction dispatch** birimi **komutları bağımlılıklarına göre execution window'a aktarır.**
- Komutlar yeniden sıralanmış bir şekilde execute edilir.



Ödev

- Çok işlemcili mimariler hakkında detaylı bir araştırma ödevi hazırlayınız.

Kaynaklar

- **TJAD70**, Tjaden G.S., Flynn M.J., "Detection and Parallel Execution of Independent. Instructions". IEEE Transactions on Computers, Vol. C-19 (October 1970), pp. 889-895.
- **KUCK72**, Kuck D.J., Muraol Y., Hen S.C., "On the Number of Opera. Lions Simultaneously Executable in Fortran-like Programs and Their resulting Speedup". IEEE Transactions on computers, Vol. C-21 (December 1972), pp. 1293-1310.
- **WEIS84**, Weiss S., Smith J.E., "Instruction Issue Logic in Pipelined Supercomputers", IEEE Transaction's on Computers, Vol. C-33 (November 1984), pp. 1013-1022.
- **ACOS86**, Acosta R.D., Kjelstrup J., Torng H.C., "An instruction issuing approach to enhancing performance in multiple functional unit processors", IEEE Transactions on Computers, v.35 n.9, p.815-828, Sept. 1986.
- **SOHI90**, Sohi G.S., "Instruction Issue Logic for High-Performance, Interrumpible, Multiple Functional
Unit, Pipelined Computers", IEEE Trans. on Computer, 39(3), pp. 349-359, 1990
- **SMIT89**, Smith J.E., "Dynamic Instruction Scheduling and the Astronautics ZS-I" IEEE Computer, July 1989, pp. 21-35.
- **JOUP89**, Jouppi N.P., Wall D.W., "Available Introduction-Level Parallelism for Superscalar and Superpipelined Machines," in ASPLOS-III, Boston, MA, April 1989.
- **LEE91**, Lee R., Kwok A., Briggs F., "The Floating Point Performance of a Superscalar SPARC Processor", Proceedings, Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, April 1991.