BM-311 Bilgisayar Mimarisi

Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Konular

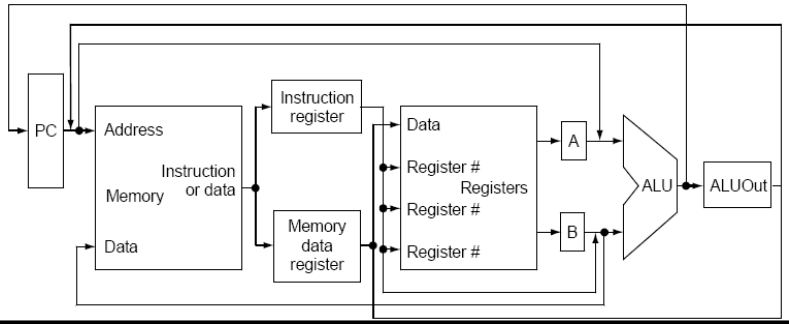
- Multicycle Oluşturmak
- Interrupt Oluşturmak

Multicycle Oluşturmak

- Multicycle çalışmada her komut birden çok adımla ifade edilir.
- Her adım bir clock cycle'da tamamlanır.
- Her komut farklı sayıda clock cycle ile gerçekleştirilebilir.
- Bir fonksiyonel birim bir komut içinde birden fazla cycle arasında paylaşılabilir.

Multicycle Oluşturmak

- Şekilde komut ve data için tek hafıza kullanılmıştır.
- Tüm toplama işlemleri için tek ALU birimi kullanılmıştır.
- Birimler arasında register'lar kullanılarak data'nın sonraki clock cycle'a aktarılması sağlanmıştır.
- Sonraki komutlarda kullanılacak data, register file, PC veya memory üzerinde saklanır.
- Aynı komutun sonraki clock cycle'ında kullanılacak data için register'lar eklenmiştir (A, B, MDR, IR, ALUOut).



Multicycle Oluşturmak

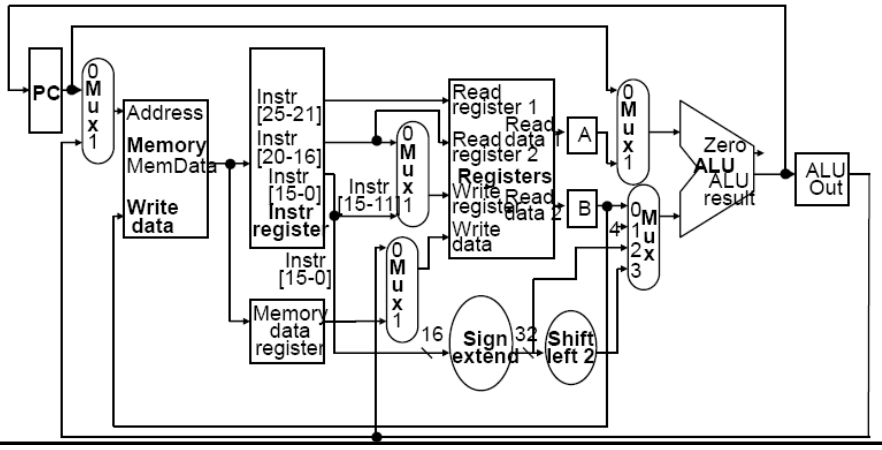
- Multicycle çalışmada her clock cycle'da aşağıdaki işlemlerden sadece bir tanesi yapılabilir:
  - Hafıza erişimi
  - Register file erişimi
  - ALU işlemi
- Data sadece aşağıdaki üç birimde üretilir ve sonraki clock cycle'da kullanılmak üzere register'larda saklanmalıdır:
  - Hafıza
  - Register file
  - ALU
- Bu işlemler için aşağıdaki register'lar eklenmiştir:
  - Instruction register (IR) ve memory data register (MDR).
  - A ve B register'ları okunan operandları saklar.
  - ALUOut register'ı ALU çıkışını saklar.

Multicycle Oluşturmak

- Fonksiyonel birimler paylaşıldığı için multiplexer'larla seçme yapılmalıdır.
- Bir hafıza birimi olduğu için PC ile ALUOut arasında seçim yapılmalıdır.
- ALU üst girişindeki seçimi yapmak için (A ile PC) multiplexer kullanılmalıdır.
- ALU alt girişinde PC+4, sign extend ve shift left seçimi için multiplexer kullanılmalıdır.
- ALU sayısı ve hafıza sayısı bire düşürülmüştür.

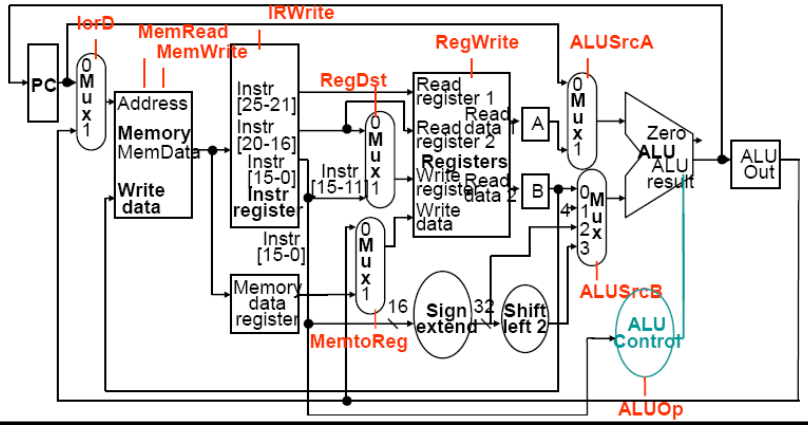
Multicycle Oluşturmak

- Şekildeki veriyolu her komut için birden fazla clock cycle gerektirdiği için ek kontrol gerekmektedir.
- 2 girişli mux için 1, 4 girişli mux için 2 kontrol işareti gereklidir.



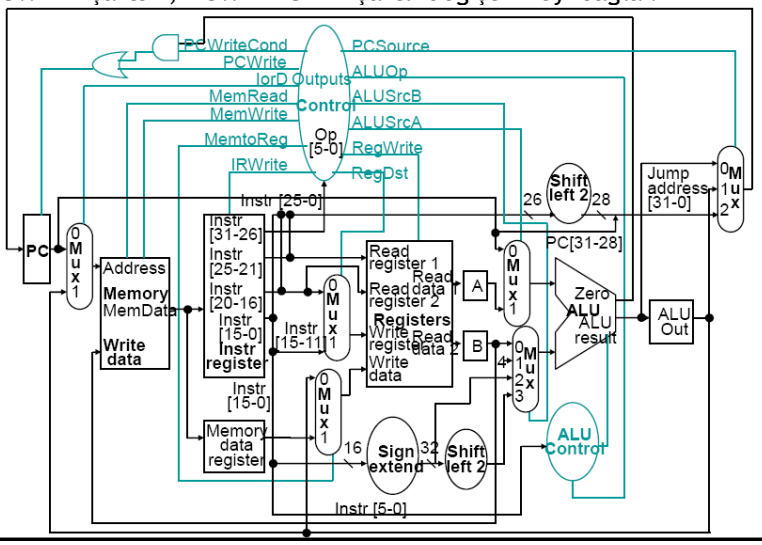
Multicycle Oluşturmak

- Branch ve jump için PC değeri aşağıdakilerden alınabilir:
  - ALU'nun hesapladığı PC+4 değeri PC'ye yazılır.
  - ALU'da hesaplanan branch target adresi PC'ye yazılır.
  - Jump için 26 bit değer 2-bit sola kaydırılır ve PC'nin artırılmış değerinin 4 bitiyle toplanır.



Multicycle Oluşturmak

- PC değeri şartlı (branch) veya şartsız (sıralı artış veya jump) yazılabilir.
- **PCWrite** şartsız, **PCWriteCond** şartlı değiştirmeyi sağlar.



Multicycle Oluşturmak

- 1-bit kontrol işaretleri aşağıdaki gibidir.

Actions of the 1-bit control signals

Signal name	Effect when deasserted	Effect when asserted
RegDst	The register file destination number for the Write register comes from the rt field.	The register file destination number for the Write register comes from the rd field.
RegWrite	None.	The general-purpose register selected by the Write register number is written with the value of the Write data input.
ALUSrcA	The first ALU operand is the PC.	The first ALU operand comes from the A register.
MemRead	None.	Content of memory at the location specified by the Address input is put on Memory data output.
MemWrite	None.	Memory contents at the location specified by the Address input is replaced by value on Write data input.
MemtoReg	The value fed to the register file Write data input comes from ALUOut.	The value fed to the register file Write data input comes from the MDR.
IorD	The PC is used to supply the address to the memory unit.	ALUOut is used to supply the address to the memory unit.
IRWrite	None.	The output of the memory is written into the IR.
PCWrite	None.	The PC is written; the source is controlled by PCSource.
PCWriteCond	None.	The PC is written if the Zero output from the ALU is also active.

Multicycle Oluşturmak

- 2-bit kontrol işaretleri aşağıdaki gibidir.

Actions of the 2-bit control signals

Signal name	Value (binary)	Effect
ALUOp	00	The ALU performs an add operation.
	01	The ALU performs a subtract operation.
	10	The funct field of the instruction determines the ALU operation.
ALUSrcB	00	The second input to the ALU comes from the B register.
	01	The second input to the ALU is the constant 4.
	10	The second input to the ALU is the sign-extended, lower 16 bits of the IR.
	11	The second input to the ALU is the sign-extended, lower 16 bits of the IR shifted left 2 bits.
PCSource	00	Output of the ALU (PC + 4) is sent to the PC for writing.
	01	The contents of ALUOut (the branch target address) are sent to the PC for writing.
	10	The jump target address (IR[25:0] shifted left 2 bits and concatenated with PC + 4[31:28]) is sent to the PC for writing.

Multicycle Oluşturmak

- Bir işlem adımlar halinde gerçekleştirilir.
- Toplama işlemi

```
Reg [Memory[PC][15:11]] <=  Reg[Memory[PC][25:21]] op
                            Reg[Memory[PC][20:16]]
```

şeklinde tanımlanır.

- Komutta 15:11 bitlerle hedef register belirlenir.
- Toplanacak iki register 25:21 ve 20:16 bitlerle belirlenir.
- Bu işlemler aşağıdaki parçalar halinde yapılabilir:
  - `IR <= Memory[PC]`
  - `A <= Reg[IR[25:21]]`
  - `B <= Reg[IR[20:16]]`
  - `ALUOut <= A op B`
  - `Reg[IR[15:11]] <= ALUOut`
  - `PC <= PC + 4`

Multicycle Oluşturmak

Tüm komutlar 5 farklı adımda gerçekleştirilebilir:

1. Instruction fetch
2. Instruction Decode ve Register Fetch
3. Execution, Memory Address Computation, Branch Completion
4. Memory Access veya R-type instruction completion
5. Memory read completion

Tüm komutlar 3-5 cycle arasında tamamlanır.

Multicycle Oluşturmak

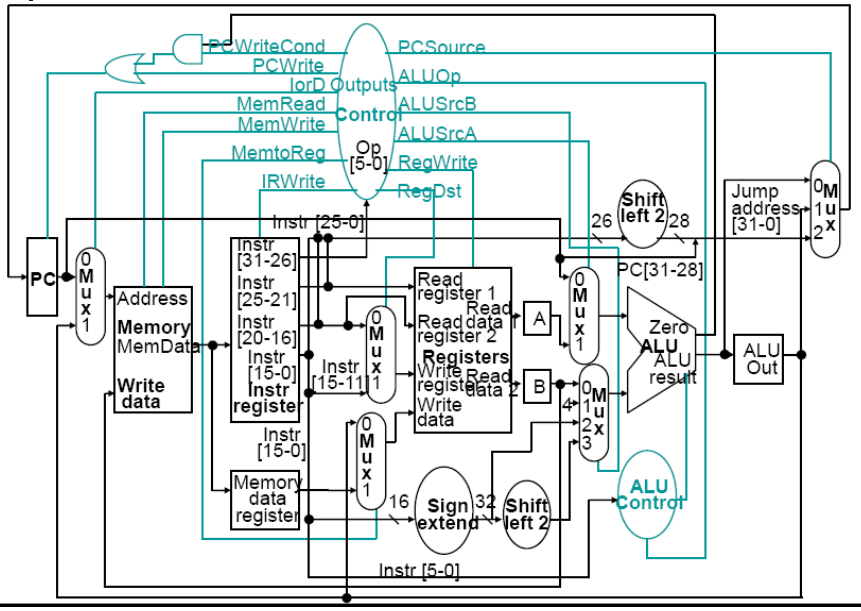
1. Instruction fetch

- IR içerisine komutun alınması için PC kullanılır.
- PC değeri 4 artırılır ve PC'ye yazılır.

```
IR <= Memory[PC];
PC <= PC + 4;
```

- MemRead ve IRWrite kontrol işaretleri etkin yapılır.
- IorD kontrol işareti 0 yapılır ve PC içeriği adres alınır.
- PC+4 işlemi için ALUSrcA işareti 0 yapılır ve PC değeri ALU'ya toplama için gönderilir.
- ALUSrcB işareti 01 yapılır ve ALU'nun diğer girişine 4 gönderilir.
- ALUOp işareti 00 yapılarak toplama seçilir.
- PCSource işareti 00 yapılır ALU çıkışı PC'ye gönderilir.
- Yazma işlemi için PCWrite etkin yapılır.

Multicycle Oluşturmak



Multicycle Oluşturmak

2. Instruction decode ve register fetch

- Önceki adımda ve şimdiki adımda komutun fonksiyonu bilinmiyor.
- Gerekiyorsa rs ve rt register'ları okunur ve A ile B'ye aktarılır.
- Branch adresi hesaplanır ve ALUOut register'ına yazılır.

```
A <= Reg[IR[25:21]];
B <= Reg[IR[20:16]];
ALUOut <= PC + (sign-extend(IR[15:0]) << 2);
```

- ALUSrcA işareti 0 ve ALUSrcB işareti 11 yapılır.
- ALUOp işareti 00 yapılır ve toplama işlemi seçilir.

Multicycle Oluşturmak

3. Execution, Memory Address Computation, Branch Completion

- Komutun işlevine göre hareket edilir.
- 4 farklı işlemden birisi gerçekleştirilir.

Memory reference

```
ALUOut <= A + sign-extend(IR[15:0]);
```

- Hafıza adresi için A register'ı ile IR[15:0] sign extend yapılarak toplanır.
- ALUSrcA işareti 1 ve ALUSrcB işareti 10 yapılır.
- ALUOp işareti 00 yapılır ve toplama işlemi seçilir.

Arithmetic-logical instruction (R-type)

```
ALUOut <= A op B;
```

- ALU iki operand üzerinde belirlenen işlemi yapar.
- ALUSrcA işareti 1 ve ALUSrcB işareti 00 yapılır.
- ALUOp işareti 10 yapılır ve function field (IR[5:0]) ile ALU kontrol işareti belirlenir.

Multicycle Oluşturmak

3. Execution, Memory Address Computation, Branch Completion

Branch

```
if (A==B) PC <= ALUOut;
```

- ALU iki register'ı karşılaştırır.
- ALU'nun zero çıkışı atlamaya karar verir.
- ALUSrcA işareti 1 ve ALUSrcB işareti 00 yapılır.
- Zero çıkışı 1 ise PCWriteCond kontrol işareti etkin yapılır.
- PCSource işareti 01 yapılır ve ALUOut register'ındaki adres PC'ye yazılır.

Jump

```
PC <= { PC[31:28], (IR[25:0]],2'b00) };
```

- PC'ye atlama adresi yazılır.
- PCWrite etkin yapılır.
- PCSource işareti 10 yapılır.

Multicycle Oluşturmak

4. Memory Access veya R-type instruction completion

- Bu adımda hafızaya erişim yapılır (load/store için).
- Hafızadan okuma yapılırsa MDR içerisine yazılır ve sonraki clock cycle'da kullanılır.

Memory reference

```
MDR <= Memory[ALUOut];    veya
Memory[ALUOut] <= B;
```

- Eğer load komutuysa hafıza okunur ve MDR'ye yazılır.
- Eğer store komutuysa hafızaya yazılır.
- ALUOut register'ına önceki clock cycle'da her iki durum içinde hafıza adresi hesaplanıp yazılmıştı.
- Store için kaynak operand B register'ındadır.
- MemRead (load için) veya MemWrite (store için) etkin yapılır.
- Hafıza adresini ALU'dan almak için IorD işareti 1 yapılır.

Multicycle Oluşturmak

4. Memory Access veya R-type instruction completion

Arithmetic-logical instruction (R-type)

```
Reg[IR[15:11]] <= ALUOut;
```

- ALUOut register'ına önceki cycle'da işlem sonucu saklanır.
- ALUOut register içeriği hedef register'a yazılacaktır.
- RegDst işareti 1 yapılır ve komut içindeki rd alanı (IR[15:11]) ile seçilen register'a yazma işlemi gerçekleştirilir.
- RegWrite etkin yapılır.
- MemtoReg işareti 0 yapılır ve ALU çıkışı yazılır.

Multicycle Oluşturmak

5. Memory read completion

- Bu adımda hafızadan alınan data register'a yazılır.

```
Reg[IR[20:16]] <= MDR;
```

- MDR register değeri önceki cycle'da hafızadan alınmıştı.
- MDR değeri register file içine yazılacak.
- MemtoReg işareti 1 yapılır.
- RegWrite etkin yapılır.
- RegDst işareti 0 yapılır ve rt (IR[20:16]) bitleriyle belirlenen register'a yazılır.

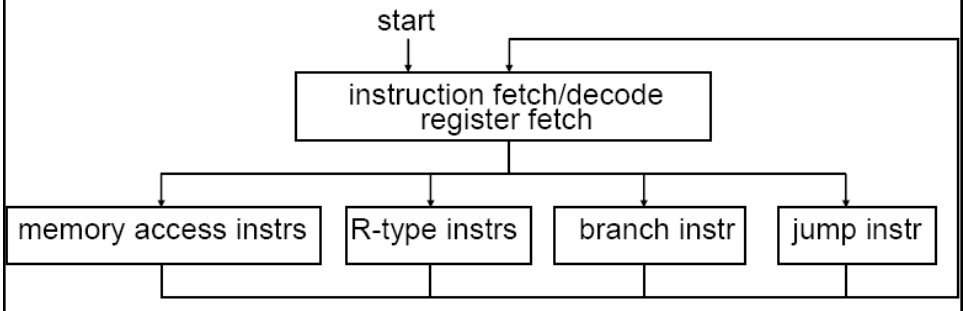
Multicycle Oluşturmak

5 adım için oluşturulan kontrol işaretleri

Step name	Action for R-type instructions	Action for memory-reference instructions	Action for branches	Action for jumps
Instruction fetch	IR <= Memory[PC] PC <= PC + 4			
Instruction decode/register fetch	A <= Reg [IR[25:21]] B <= Reg [IR[20:16]] ALUOut <= PC + (sign-extend (IR[15:0]) << 2)			
Execution, address computation, branch/jump completion	ALUOut <= A op B	ALUOut <= A + sign-extend (IR[15:0])	If (A == B) PC <= ALUOut	PC <= {PC [31:28], (IR[25:0]),2'b00)}
Memory access or R-type completion	Reg [IR[15:11]] <= ALUOut	Load: MDR <= Memory[ALUOut] or Store: Memory [ALUOut] <= B		
Memory read completion		Load: Reg[IR[20:16]] <= MDR		

Multicycle Oluşturmak

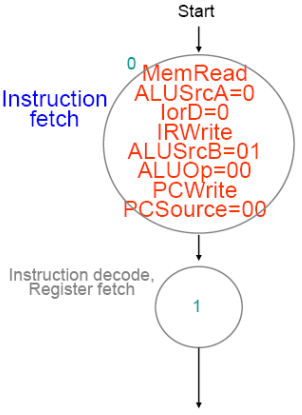
Şekilde her birisi 1 clock cycle sürede tamamlanan 5 adımla programın çalışması görülmektedir.

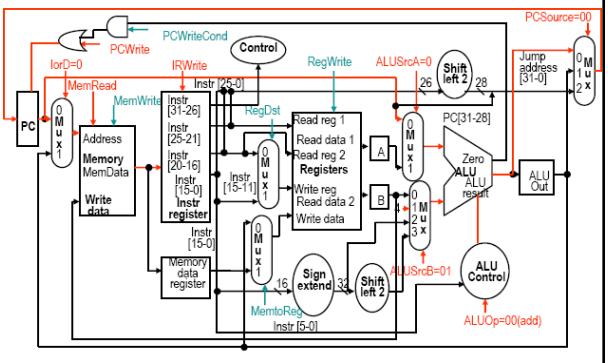


Her adım bir sonlu durum makinesiyle (finite state machine) tasarlanabilir.

Multicycle Oluşturmak

State 0: Instruction fetch

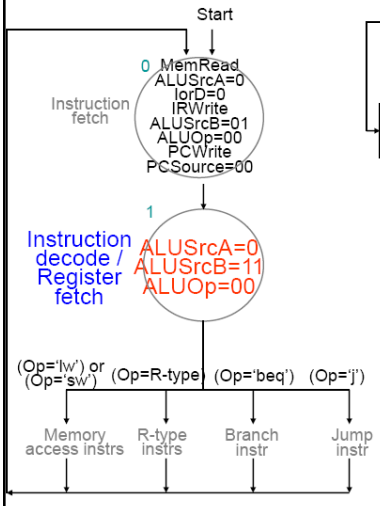


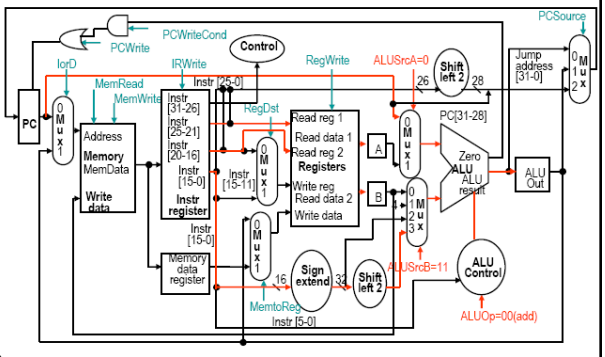



```
IR <= Memory[PC];
PC <= PC + 4;
```

Multicycle Oluşturmak

State 1: Instruction decode/register fetch

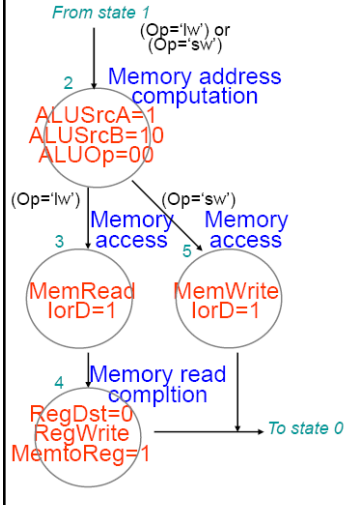


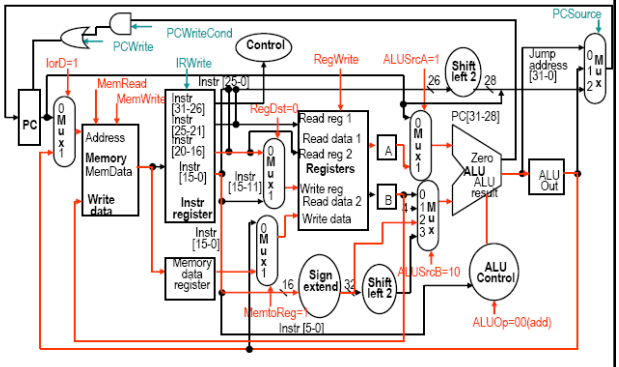



```
A <= Reg[IR[25:21]];
B <= Reg[IR[20:16]];
ALUOut <= PC + (sign-extend(IR[15:0]) << 2);
```

Multicycle Oluşturmak

Memory erişim komutları

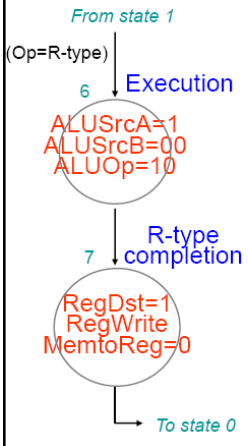


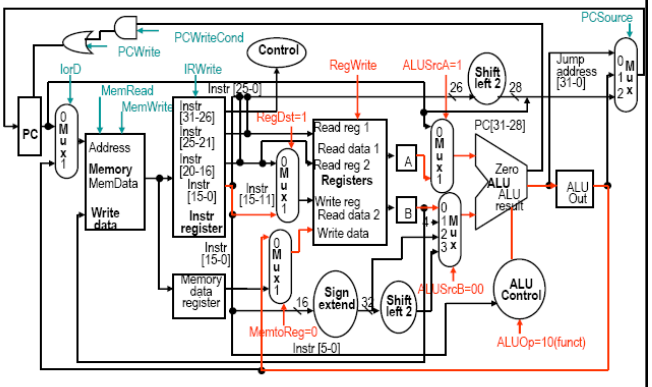



```
ALUOut <= A + sign-extend(IR[15:0]);

MDR <= Memory[ALUOut];     or
Memory[ALUOut] <= B;

Reg[IR[20:16]] <= MDR;
```

Multicycle Oluşturmak

R-type komutlar

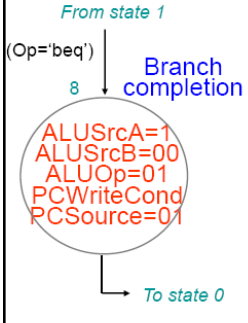


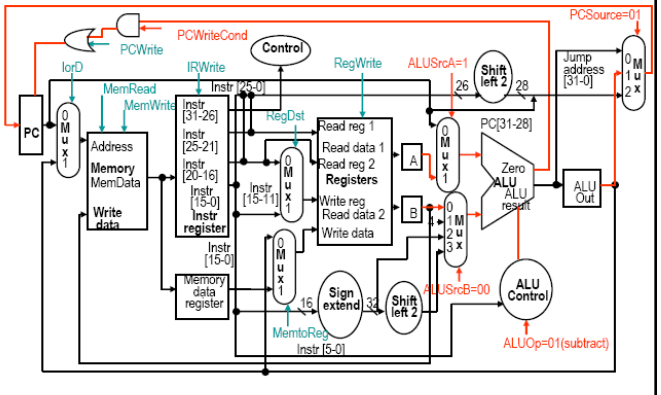



```
ALUOut <= A op B;

Reg[IR[15:11]] <= ALUOut;
```

Multicycle Oluşturmak

Branch komutu

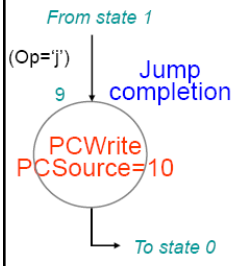


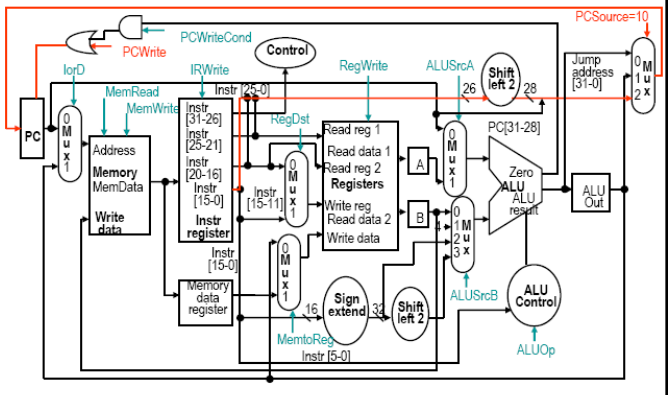



```
if (A==B) PC <= ALUOut;
```

Multicycle Oluşturmak

Jump komutu

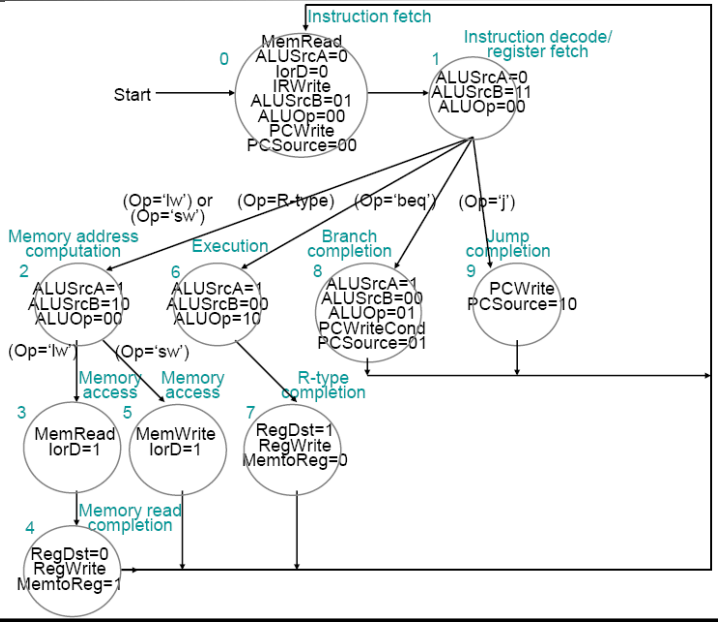





```
PC <= { PC[31:28], (IR[25:0]],2'b00) };
```

Multicycle Oluşturmak

Finite State Machine ile komple gösterim



Konular

- Multicycle Oluşturmak
- Interrupt Oluşturmak

Interrupt Oluşturmak

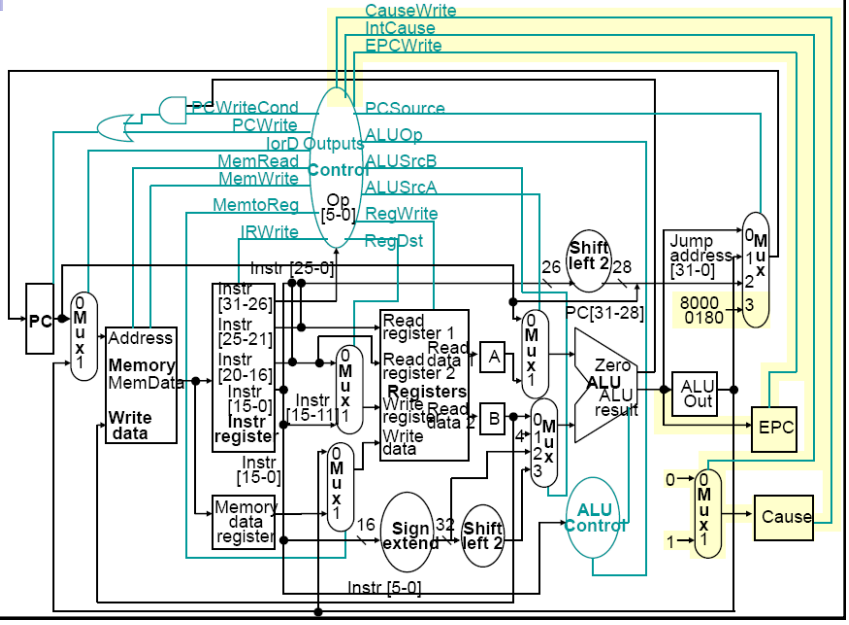
- İşlemci tasarımının en önemli kısmı kontrol birimi tasarımıdır.
- Kontrol biriminin en önemli tasarım kısmı ise interrupt'lardır.
- Interrupt'lar ve Exception'lar program akışını branch ve jump gibi değiştirirler.
- Arithmetic overflow bir exception'dır.
- I/O cihazları interrupt oluşturur.
- Interrupt oluştuğunda bulunulan adres EPC (Exception Program Counter) register'ına yazılır ve kontrol OS (Operating System)'ye bırakılır.
- Burda tanımlanmamış bir komut çalıştırmayla aritmetik taşma için gerekli tasarım yapılacaktır.
- EPC programa tekrar dönüldüğünde çalışacak komutun adresini saklar.

Interrupt Oluşturmak

- OS bir exception oluştuğunda nedenini ve hangi komutun oluşturduğunu bilmek ister.
- Cause register veya interrupt vector table kullanılır.
- Tasarıma iki register eklenmiştir (EPC ve Cause register)
- EPC etkilenen komutun adresini saklar.
- Cause register ise nedenini saklar (0-undefined instruction, 1-arithmetic overflow).
- İki kontrol işareti eklenmiştir (EPCWrite ve CauseWrite).
- IntCause kontrol işareti Cause register'ının en sağdaki bitini değiştirir.
- Exception oluştuğunda gidilecek adres 8000 0180H olarak verilmiştir.
- PCSource işareti 11 olduğunda exception adresi seçilir.

Interrupt Oluşturmak

Komple tasarım yandaki gibidir.



Interrupt Oluşturmak

FSM ile komple sistem