

Bilgisayar Organizasyonu

Ismail Kadayif
Canakkale Onsekiz Mart Universitesi
Bilgisayar Muhendisligi

4/13/2004

Formal Diller

1.1

<http://www.ecs.umass.edu/ece/vspgroup/burleson/courses/232/spim/>

4/13/2004

Bilgisayar Organizasyonu

1.2

Programlama

- programlama dilleri
 - high-level languages (yuksek seviyeli diller) (C, C++, Java, Fortran, ...)
 - kompleks operationlar ifade edilebilir
 - programlama islemini kolaylastirir
 - low-level languages (alcak seviyeli diller) (C, assembly, makine dili, ...)
 - operationlari basit (program yazimi zor ve kolayca hata yapilmaya musait)

4/13/2004

Bilgisayar Organizasyonu

1.3

Programlama

```
hypotenuse := sqrt (sqr(a) + sqr(b));
```



```
mul  asquare, a, b
mul  bsquare, b, b
mdd  sumsquare, asquare, bsquare
sqrt hypotenuse, sumsquare
```

4/13/2004

Bilgisayar Organizasyonu

1.4

Programlama

```
hypotenuse := sqrt (sqr(a) + sqr(b));
```



```
l.s  $f0, aa
mul.s $f0,$f0,$f0
l.s  $f2, bb
mul.s $f2,$f2,$f2
add.s $f4,$f0,$f2
s.s  $f4, sumsquare
neg.s $f6,$f4
s.s  $f6, hypotenuse
```

4/13/2004

Bilgisayar Organizasyonu

1.5

Programlama

```
0011 1100 0000 0001 0001 0000 0000 0000
1100 0100 0010 0000 0000 0000 0000 0000
0100 0110 0000 0000 0000 0000 0000 0010
.....
0011 1100 0000 0001 0001 0000 0000 0000
```

4/13/2004

Bilgisayar Organizasyonu

1.6

Compilation



4/13/2004

Bilgisayar Organizasyonu

1.7

Bilgisayar Mimarisi

- assembly dilinde programlama bilgisayar mimarisine bağlı olarak değişiklik gösterir (instruction set architecture) (ISA)
- ISA az sayıda bir operationlardan (instruction) oluşur. Instructionlar hardware tarafından doğrudan çalıştırılabilirler.
- bilgisayarın machine language üreticisi tarafından belirlenir
- MIPS RISC architecture

4/13/2004

Bilgisayar Organizasyonu

1.8

Low-level Languages (MIPS mimarisinde)

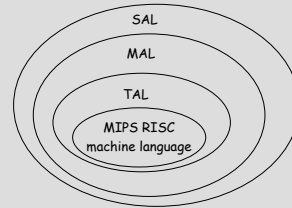
- True Assembly Language (TAL): ISA'daki instructionslardan oluşur.
 - TAL daha aşağı seviyede programlama yapılamaz
- More Abstract Language (MAL) (MIPS Assembly Language)
 - MIPS RISC assembler, MAL programı önce TAL programı, daha sonra TAL'daki her instructionu MIPS RISC makine koduna çevirir
- MAL programları TAL programları kadar efficienttir.
- MAL'da programlama TAL'da programlamadan daha kolay
- Simple Abstract Language (SAL). MAL'dan daha yüksek seviyedir. High-level language'lerle Assembly Language arasındaki boşluğu doldurmak için kullanılır.

4/13/2004

Bilgisayar Organizasyonu

1.9

MIPS RISC mimarisindeki abstraction



4/13/2004

Bilgisayar Organizasyonu

1.10

Bilgisayar Organizasyonu

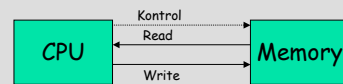
- programın çalışması sırasında, processor instructionları run eder.
 - instructionlar değişkenlere nasıl değerler atandığını (memory) belirler.
- memorynin boyu sabit fakat oldukça geniştir.
 - programda tanımlanan her bir variable (değişkene) karşı bir bellek yeri (memory location) tahsis edilir.
 - herhangi bir değişkenin değerinin okunması/değişkene değer atanması, bu değişkenin bulunduğu bellek yerinin okunması/yazılması anlamına gelir (load/store).
 - **binding**: bellek yerinin bir değişkene tahsis edilmesine

4/13/2004

Bilgisayar Organizasyonu

1.11

Bilgisayar Organizasyonu




CPU (Central Processing Unit):

4/13/2004

Bilgisayar Organizasyonu

1.12



```

part1:      x1:=x2+x3;
            x4:=x2-x5;
            if(x4=x1) then goto 10;
            x1:=x2+x5;


10:

part1:      add    x1, x2, x3
            sub    x4, x2, x5
            beq    x4, x1, part2
            add    x1, x1, x5

part2:

```

4/13/2004 Bilgisayar Organizasyonu 1.13



Program Execution Sirasinda Temel Adimlar

1. instruction fetch
2. program counter (PC) update
3. instruction decode
4. operand load
5. operation execution
6. storage of results (sonuclarin bellge yazilmasi)

4/13/2004 Bilgisayar Organizasyonu 1.14



SAL da Programlama

```


program summation(input, output);
var
    n:      integer;
    sum:    integer;
    i:      integer;

begin
    write('Please Enter a positive integer: ');
    readln(n);
    writeln;
    sum:=0;
    for i:=0 to n do
        sum:=sum+i;
    write('The sum of the first '); writeln(n);
    write(' integers is '); writeln(sum);

end.

```

4/13/2004 Bilgisayar Organizasyonu 1.15




```

.data
str1: .ascii "Please enter a positive integer: "
str2: .ascii "The sum of the first "
str3: .ascii " integers is "
newline: .byte '\n'

n: .word 0
sum: .word 0
i: .word 0
temp: .word

```

4/13/2004 Bilgisayar Organizasyonu 1.16



```


__start: puts str1
        get  n
        put  newline

for:     sub    temp, n, i
        bltz   temp, endfor
        add    sum, sum, i
        add    i, i, 1
        b      for

endfor:  puts str2
        put    n
        puts str3
        put    sum
        put    newline
done

```

4/13/2004 Bilgisayar Organizasyonu 1.17



Branch Instructions (SAL)

b	label	# goto label
beq	x, y, label	# if x = y then goto label
bne	x, y, label	# if x <> y then goto label
blt	x, y, label	# if x < y then goto label
bgt	x, y, label	# if x > y then goto label
ble	x, y, label	# if x <= y then goto label
bge	x, y, label	# if x >= y then goto label

4/13/2004 Bilgisayar Organizasyonu 1.18

Branch Instructions (SAL)

```
bltz  x, label    # if x < 0 then goto label
bgtz  x, label    # if x > 0 then goto label
blez  x, label    # if x <= 0 then goto label
bgez  x, label    # if x >= 0 then goto label
beqz  x, label    # if x = 0 then goto label
bnez  x, label    # if x <> 0 then goto label
```

4/13/2004

Bilgisayar Organizasyonu

1.19

Communication Instructions

```
get  x    # readln(x) (x .word veya .float)
get  x    # read(x) (x .byte)
put  x    # write(x) (x, .word, .float, .byte)
puts stringname # write(string) (.asciiiz)
```

4/13/2004

Bilgisayar Organizasyonu

1.20

```
if (A > 0) then
  B := C div A
else
  B := A+10;
```

```
        blez  A, elsepart
        div   B, C, A
        b     endif
elsepart: add   B, A, 10
endif:
```

4/13/2004

Bilgisayar Organizasyonu

1.21

```
        bgtz  A, ifpart
        add   B, A, 10
        b     endif
ifpart:  div   B, C, A
endif:
```

4/13/2004

Bilgisayar Organizasyonu

1.22

```
if ((A = B) or (C < D)) then
begin
  A := A + 1;
  B := B - 1;
  D := A + C;
end;
```

```
        beq   A, B, do_if
        blt   C, D, do_if
        b     end_if
do_if:  add   A, A, 1
        add   B, B, -1
        add   D, A, C
endif:
```

4/13/2004

Bilgisayar Organizasyonu

1.23


```
if ((A = B) and (C = D) or (E < 0)) then
begin
  A := A + 1;
  C := E;
end;
```

```
        bne   A, B, check_E
        beq   C, D, do_if
        bgez  E, end_if
check_E: add   A, A, 1
do_if:  add   D, A, C
end_if:
```

4/13/2004

Bilgisayar Organizasyonu

1.24



```

result := 1;
counter := exponent;
while (counter > 0 ) do
begin
    result := result * base
    counter := counter - 1;
end;


```

```

while:      move    result, 1
            move    counter, exponent
            blez    counter, endwhile
            mul     result, base
            sub     counter, counter, 1
            while
endwhile:

```


4/13/2004 Bilgisayar Organizasyonu 1.25



Procedure

- Programın modüler olmasına yardımcı olurlar.
- Procedure kullanımı
 - SAL sınırlamaları
 - Procedure parametre aktarma olanagi yoktur
 - Recursive procedure cagirma olanagi yoktur
- Temel adimlar
 - Save return address
 - Procedure call
 - Execute procedure
 - Return

4/13/2004 Bilgisayar Organizasyonu 1.26



```

.text
.
.
.
call:      b proc
rtaddr:    .
.
.

```

procedure call


return location

```

proc:      # procedure code here
.
.
.
b          rtaddr

```

4/13/2004 Bilgisayar Organizasyonu 1.27



```

la        proc1_ret, ret_addr1
b         proc1
return_addr1:

```

```

la        proc1_ret, ret_addr2
b         proc1
return_addr2:

```

```

proc1:    # procedure code here
.
.
.
b         (proc1_ret)

```

4/13/2004 Bilgisayar Organizasyonu 1.28