

MODUL 6 : PCSPIM DAN BAHASA ASSEMBLY MIPS (Bagian-2)

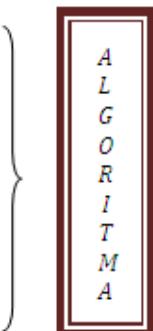
Tujuan KerjaLab:

1. Mengetahui dan memahami sintaks-sintaks SPIM 2
2. Mengimplementasikan bahasa assembly MIPS dalam studi kasus yang lebih mendalam

❖ Penggunaan SPIM dipakai pada saat kondisional (if ...then ...else...)

Kondisional kali ini akan diterapkan pada kasus pengecekan bilangan yang diinputkan user, apakah bilangan tersebut merupakan bilangan ganjil atau bilangan genap.

```
Output( "Masukkan nilai:" );
If (nilai mod 2 == 0)then
    Output( "genap" );
Else
    Output( "ganjil" )
```



The code is grouped into two main sections by curly braces. The first section starts with 'Output("Masukkan nilai:")' and ends with 'Output("genap")'. The second section starts with 'Else' and ends with 'Output("ganjil")'. To the right of the code, there is a vertical stack of letters enclosed in a rectangular border, arranged from top to bottom as A, L, G, O, R, I, T, M, A. This likely represents memory addresses or labels used in the assembly code.

Bila algoritma diatas diubah kedalam bentuk SPIM maka hasilnya

MODUL 6 : PCSPIM DAN BAHASA ASSEMBLY MIPS (Bagian-2)

```
.data
msg1: .asciiz "Masukkan nilai: "
msg2: .asciiz "Genap"
msg3: .asciiz "Ganjil"
.text
main:
li $v0, 4                  #menampilkan pesan msg1
la $a0,msg1
syscall
li $v0,5                  #baca inputan user
syscall
add $t0,$t0,$v0
addi $t1,$t1,2
div $t0,$t1
mfhi $t2
beq $t2, $zero,genap
la $a0,msg3
li $v0,4                  #jika nol ke genap
#keluarkan kata "ganjil"
#keluar
j exit
genap:
la $a0,msg2
li $v0,4
syscall
```

Printscreen:

Status = JUUUtt

R0 (r0) = 00000000
R1 (at) = 10010000

Masukkan nilai: 10
Genap

DATA

Memory and register

Registers:

| | |
|------|------------|
| \$t0 | = 00000000 |
| \$t1 | = 00000000 |
| \$t2 | = 00000000 |
| \$a0 | = 0(\$sp) |
| \$a1 | \$sp |
| \$a2 | \$a1 |
| v0 | \$a0 2 |
| \$a2 | \$a2 9 |

Memory:

| | |
|-----------------------------|----------|
| [0x00400000] | 00000000 |
| [0x00400004] | 00000000 |
| [0x00400008] | 00000000 |
| [0x0040000c] | 00000000 |
| [0x00400010] | 00000000 |
| [0x10000000]...[0x10010000] | Ba6961 |
| [0x10010010] | 00006c |
| [0x10010020]...[0x10010030] | |

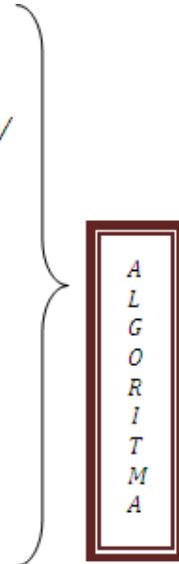
MODUL 6 : PCSPIM DAN BAHASA ASSEMBLY MIPS

(Bagian-2)

- ❖ Penggunaan SPIM untuk bilangan Prima

Pada contoh kali ini kita akan mengecek apakah inputan user merupakan bilangan prima atau bukan, dan juga kita akan mengimplementasikan perulangan (for...) beserta kondisional (if...then...else).

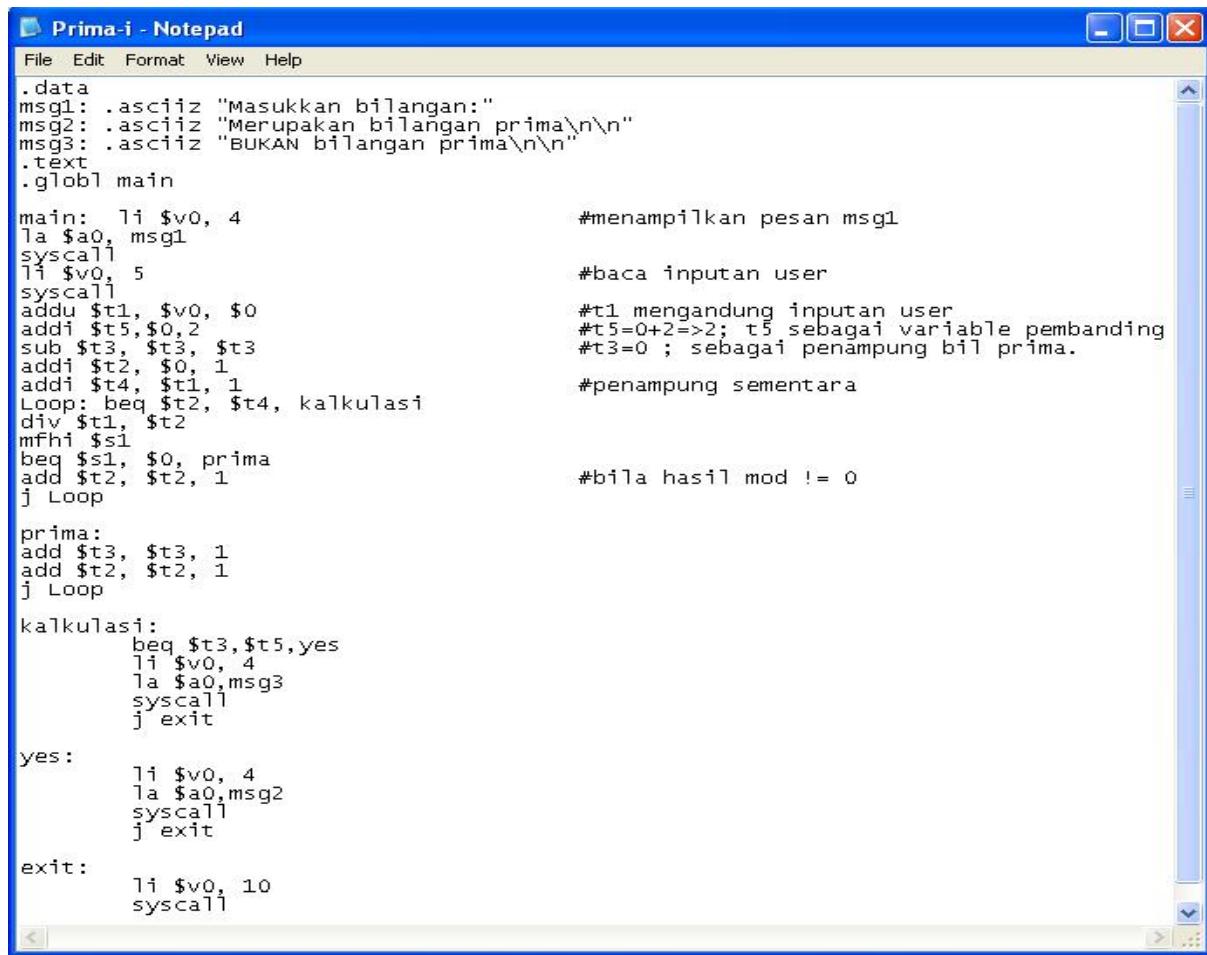
```
int x,i,p;  
  
output="Masukan bilangan:";  
read x; /* x adalah inputan user */  
  
p=0;  
For (i=1 ; i<=x ; i++)  
{  
    If x mod i ==0 then  
        p=p+1;  
}  
  
If p==2 then  
    Output="Merupakan bilangan Prima"  
Else  
    Output="BUKAN bilangan Prima";
```



Bila diconvert menjadi SPIM syntax maka perlu kita ketahui bahwa,

```
t1 = x (inputan user);  
t2 = i;  
t3 = p;  
t4 = penampung nilai (x+1) ;  
t5 = penampung bilangan 2;  
s1= hasil dari perhitungan (x mod i)
```

MODUL 6 : PCSPIM DAN BAHASA ASSEMBLY MIPS (Bagian-2)



The screenshot shows a Windows Notepad window with the title "Prima-i - Notepad". The window contains the following MIPS assembly code:

```
.data
msg1: .asciiz "Masukkan bilangan:"
msg2: .asciiz "Merupakan bilangan prima\n\n"
msg3: .asciiz "BUKAN bilangan prima\n\n"
.text
.globl main
main: li $v0, 4                  #menampilkan pesan msg1
la $a0, msg1
syscall
li $v0, 5                      #baca inputan user
syscall
addu $t1, $v0, $0
addi $t5,$0,2
sub $t3, $t3, $t3
addi $t2, $0, 1
addi $t4, $t1, 1
Loop: beq $t2, $t4, kalkulasi
div $t1, $t2
mfhi $s1
beq $s1, $0, prima
add $t2, $t2, 1
j Loop
prima:
add $t3, $t3, 1
add $t2, $t2, 1
j Loop
kalkulasi:
beq $t3,$t5, yes
li $v0, 4
la $a0, msg3
syscall
j exit
yes:
li $v0, 4
la $a0, msg2
syscall
j exit
exit:
li $v0, 10
syscall
```

MODUL 6 : PCSPIM DAN BAHASA ASSEMBLY MIPS (Bagian-2)

Printscreen:

The screenshot shows the PCSPIM debugger interface. The console window displays the following output:

```
Masukkan bilangan:5
PC      = 00000000
Status   = 3000ff10
R0 (r0) = 00000000
R1 (at) = 10010000

[0x00400000] 0x8
[0x00400004] 0x2
[0x00400008] 0x2
[0x0040000c] 0x0
[0x00400010] 0x0

DATA
[0x10000000] ... [0x]
[0x10010000]
r0=10010000
```

The registers window shows:

| | |
|---------|------------|
| PC | = 00000000 |
| Status | = 3000ff10 |
| R0 (r0) | = 00000000 |
| R1 (at) | = 10010000 |

The memory dump window shows the memory starting at address 0x10000000.

The stack dump window shows the stack starting at address \$sp (0x00400000). It contains the following data:

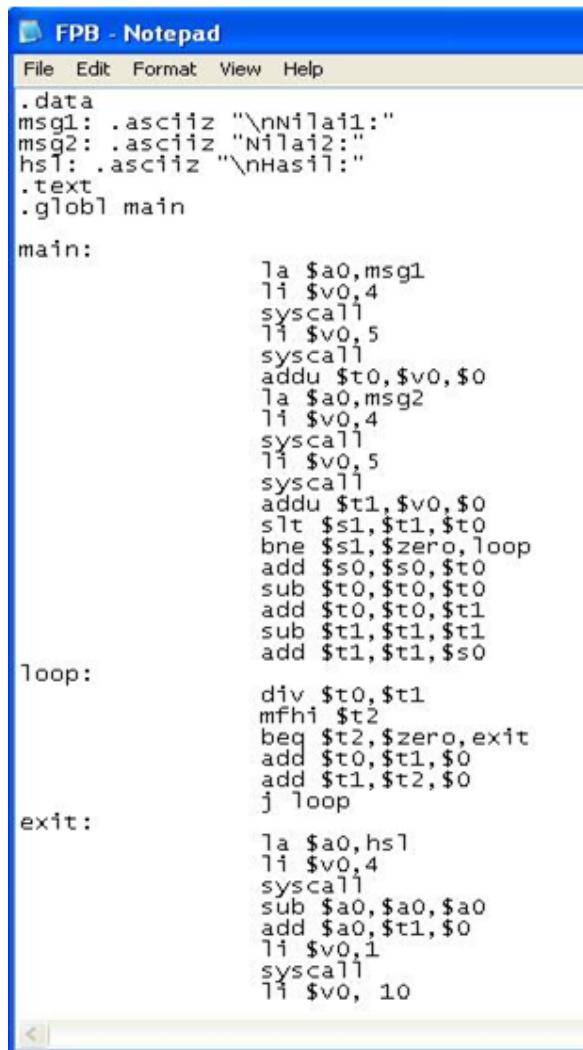
| | | |
|------|------|-----|
| a1 | \$sp | 4 |
| a2 | \$a1 | 4 |
| \$a0 | 2 | |
| 2 | \$a2 | \$v |

At the bottom of the stack dump, there are two values: 6e61 and 6220.

❖ Penggunaan SPIM untuk menghitung FPB

Pada contoh kali ini kita akan mengecek apakah inputan user merupakan bilangan FPB atau bukan.

MODUL 6 : PCSPIM DAN BAHASA ASSEMBLY MIPS (Bagian-2)



The screenshot shows a Windows-style Notepad window titled "FPB - Notepad". The menu bar includes File, Edit, Format, View, and Help. The main content area contains the following MIPS assembly code:

```
.data
msg1: .asciiz "\nNilai1:"
msg2: .asciiz "Nilai2:"
hs1: .asciiz "\nHasil:"
.text
.globl main

main:
    la $a0,msg1
    li $v0,4
    syscall
    li $v0,5
    syscall
    addu $t0,$v0,$0
    la $a0,msg2
    li $v0,4
    syscall
    li $v0,5
    syscall
    addu $t1,$v0,$0
    slt $s1,$t1,$t0
    bne $s1,$zero,loop
    add $s0,$s0,$t0
    sub $t0,$t0,$t0
    add $t0,$t0,$t1
    sub $t1,$t1,$t1
    add $t1,$t1,$s0
loop:
    div $t0,$t1
    mfhi $t2
    beq $t2,$zero,exit
    add $t0,$t1,$0
    add $t1,$t2,$0
    j loop
exit:
    la $a0,hs1
    li $v0,4
    syscall
    sub $a0,$a0,$a0
    add $a0,$t1,$0
    li $v0,1
    syscall
    li $v0, 10
```

MODUL 6 : PCSPIM DAN BAHASA ASSEMBLY MIPS (Bagian-2)

Printscreen:

The screenshot shows the PCSPIM debugger interface with the following details:

- Registers:** Shows \$t0 = 00000000, \$t1 = Nilai1:10, \$t2 = Nilai2:20, \$r0 = 00000000, \$at = Hasil:10, \$a1 = 10010000.
- Memory Dump:** A table showing memory locations from \$00400000 to \$00400010. The values are: \$00400000: 0x8, \$00400004: 0x2, \$00400008: 0x2, \$0040000c: 0x0, \$00400010: 0x0.
- Stack Dump:** Shows the stack starting at \$0(\$ssp). The stack contents are: a1 \$sp, a2 \$a1, \$a0 2, 2 \$a2 \$.
- DATA Section:** Shows the DATA section with memory locations \$10000000 to \$10010010. The values are: \$10000000...[0x1], \$10010000, \$10010010, \$100100201.
- Registers:** Shows the \$r0 register with value 6961 and the \$at register with value 0000.