LAPORAN PRAKTIKUM 2 – SISTEM OPERASI

KH003

Disusun untuk Memenuhi Tugas Praktikum Sistem Operasi

Dosen Pengampu: Adi Widianto, S.Kom, M.Kom

Devina Dewi Faustina Basam

20220801250

PROGRAM STUDI TEKNIK INFORMATIKA

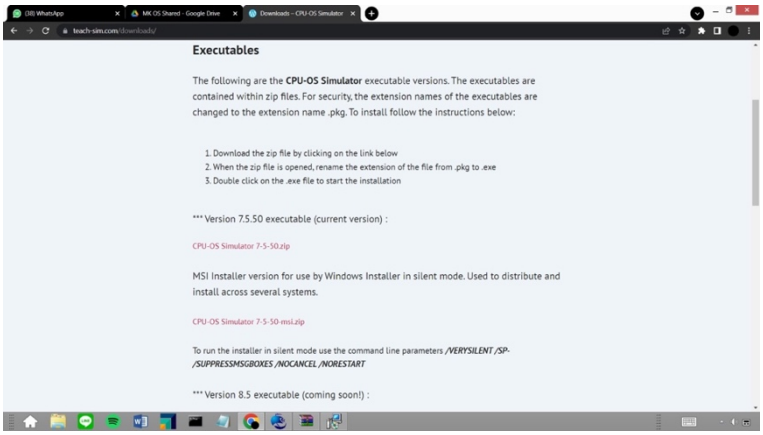
FAKULTAS ILMU KOMPUTER

UNIVERSITAS ESA UNGGUL

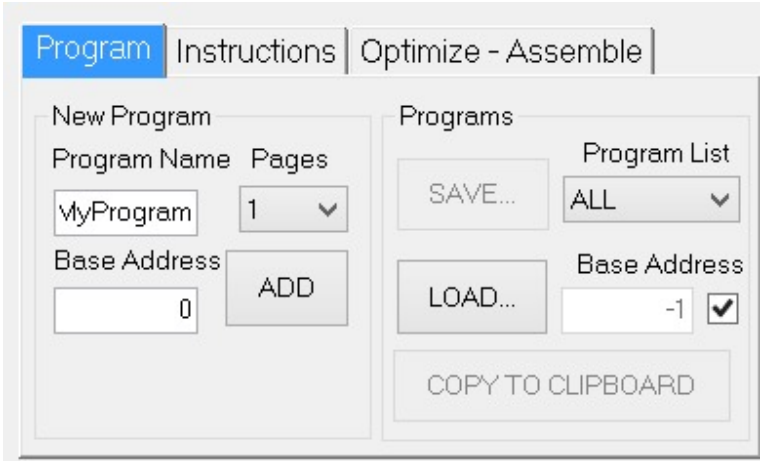
2023

PERTEMUAN 8

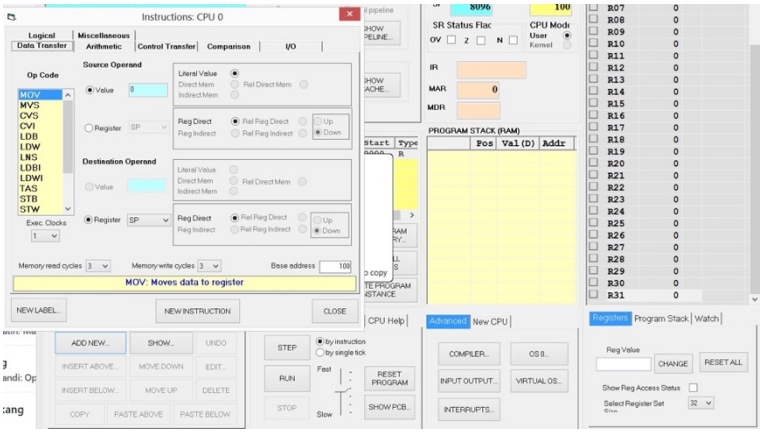
1. Download CPU OS SIMULATOR

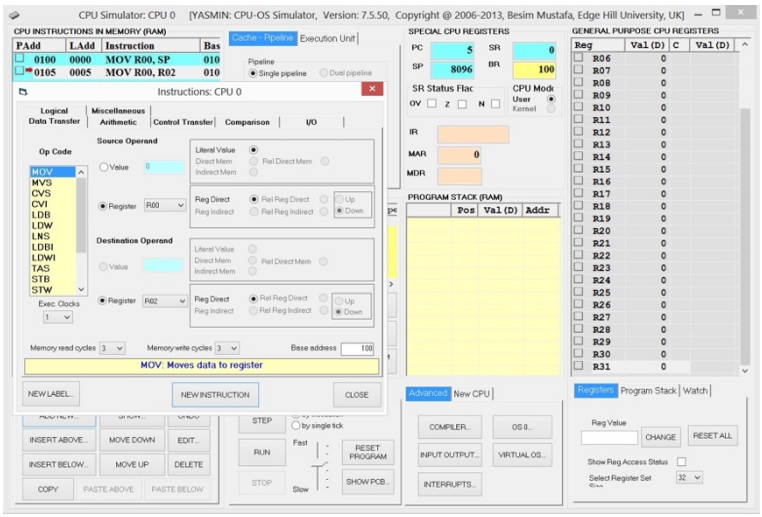


2. Membuat program baru



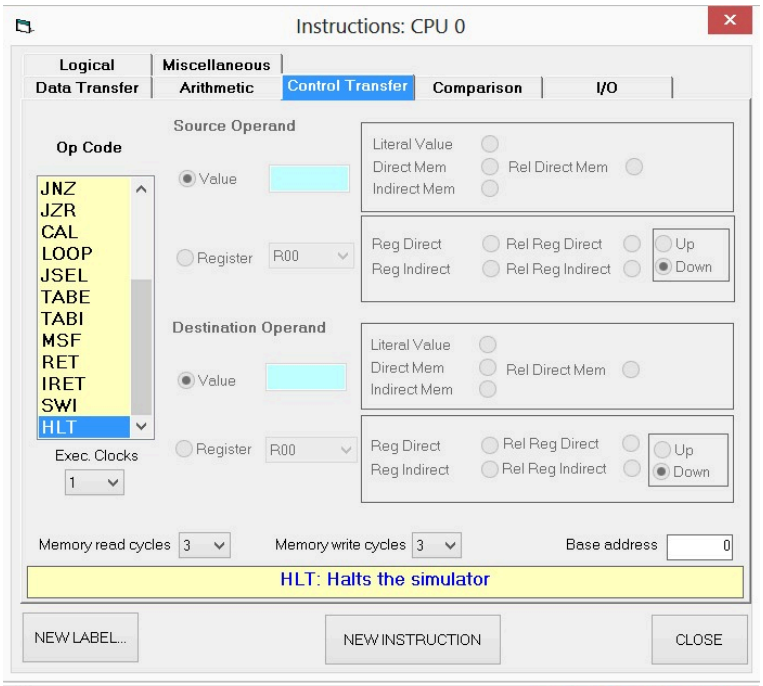
3. Membuat instruksi mov

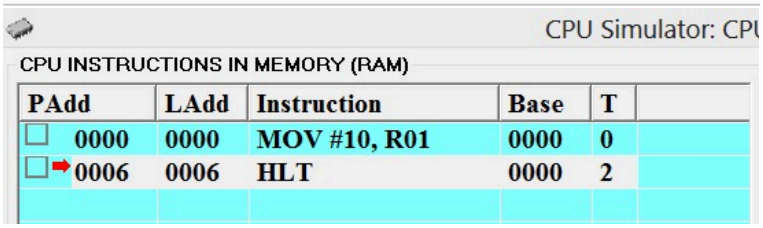



CPU Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright @ 2006-2013, Besim Mustafa, Edge Hill University, UK]
CPU INSTRUCTIONS IN MEMORY (RAM)
PAdd LAdd Instruction Bas
0100 0000 MOV R00, SP 010
0105 0005 MOV R00, R02 010
Cache - Pipeline
Execution Unit
Pipeline
Single pipeline
Dual pipeline
Instructions: CPU 0
Logical Data Transfer
Miscellaneous Arithmetic
Control Transfer
Comparison
I/O
Op Code
MOV
MVS
CVS
CVI
LDB
LDW
LNS
LDBI
LDWI
TAS
STB
STW
Source Operand
Value
Register
R00
Literal Value
Direct Mem
Indirect Mem
Rel Direct Mem
Reg Direct
Reg Indirect
Rel Reg Direct
Rel Reg Indirect
Up
Down
Destination Operand
Exec. Clocks
Memory read cycles
Memory write cycles
Base address
MOV: Moves data to register
NEW LABEL...
NEW INSTRUCTION
CLOSE
INSERT ABOVE...
MOVE DOWN
EDIT...
INSERT BELOW...
MOVE UP
DELETE
COPY
PASTE ABOVE
PASTE BELOW
STEP
by single tick
RUN
STOP
Fast
Slow
RESET PROGRAM
SHOW PCB...
SPECIAL CPU REGISTERS
PC 5
SR 0
SP 8096
BR 100
SR Status Flac
OV Z N
CPU Mode
User
Kernel
IR
MAR 0
MDR
PROGRAM STACK (RAM)
Pos Val (D) Addr
GENERAL PURPOSE CPU REGISTERS
Reg Val (D) C Val (D)
R06 0
R07 0
R08 0
R09 0
R10 0
R11 0
R12 0
R13 0
R14 0
R15 0
R16 0
R17 0
R18 0
R19 0
R20 0
R21 0
R22 0
R23 0
R24 0
R25 0
R26 0
R27 0
R28 0
R29 0
R30 0
R31 0
Advanced
New CPU
COMPILER...
OS 0...
INPUT OUTPUT...
VIRTUAL OS...
INTERRUPTS...
Registers
Program Stack
Watch
Reg Value
CHANGE
RESET ALL
Show Reg Access Status
Select Register Set
32

PERTEMUAN 9

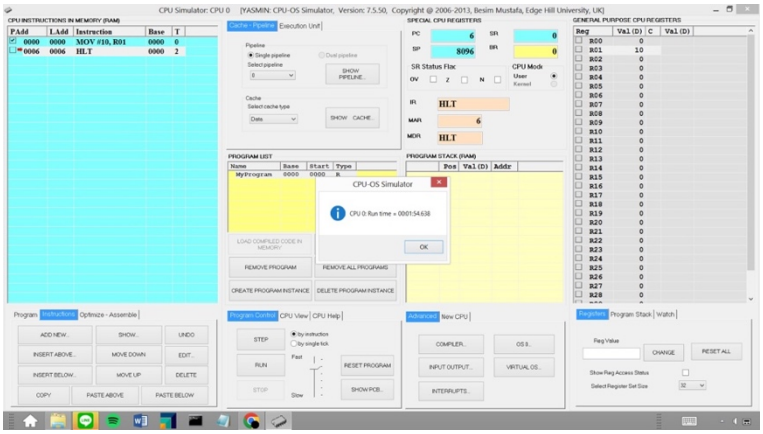
1. Membuat instruksi dan diakhiri HLT



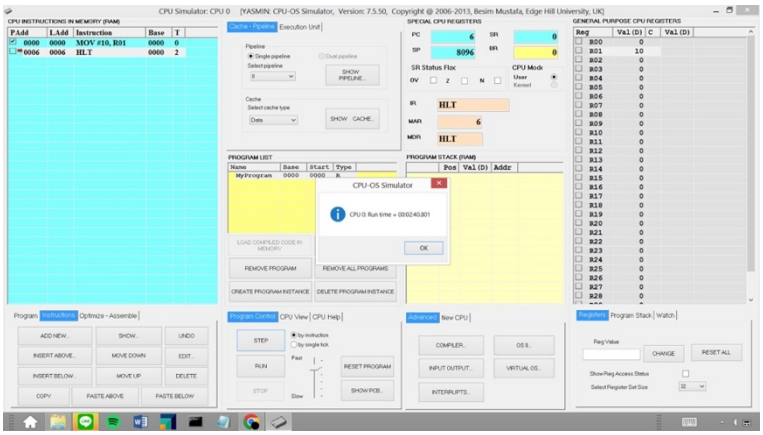


2. cara menjalankan instruksi

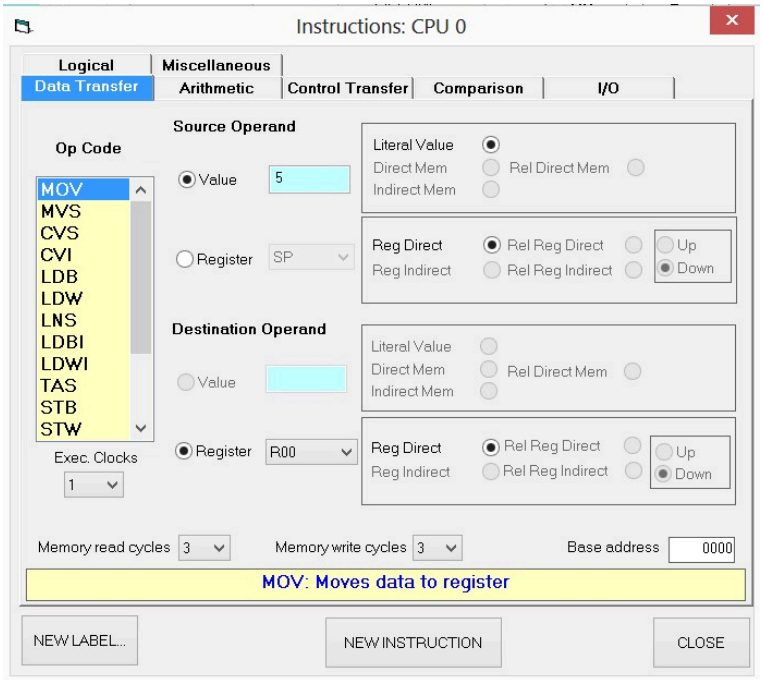
Double klik



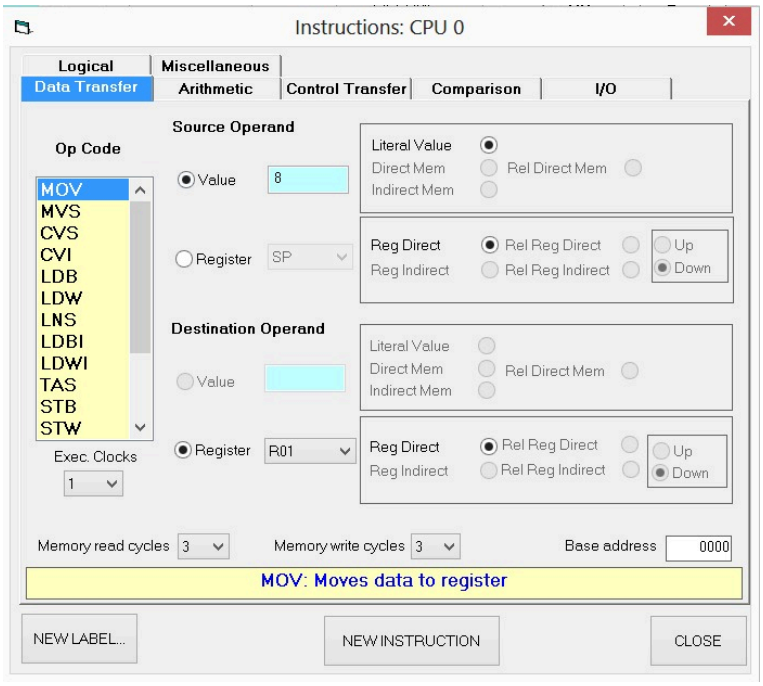
Step



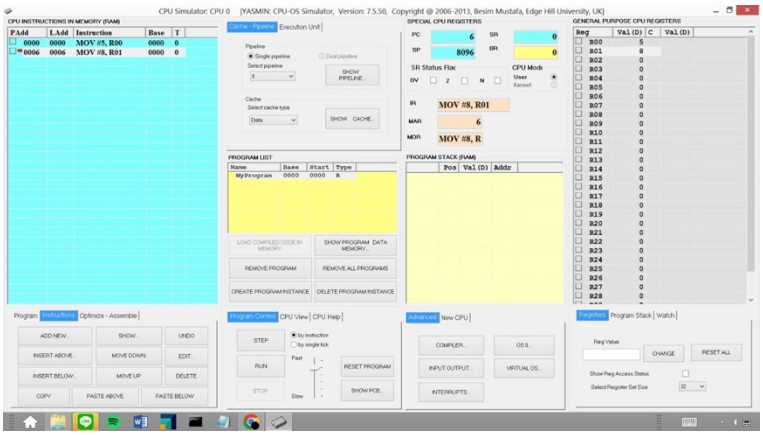
3. LATIHAN
    a. Transfer data
        - Instruksi yang memindahkan (move) angka 5 ke register R00.



        - Instruksi yang memindahkan (move) angka 8 ke register R01.

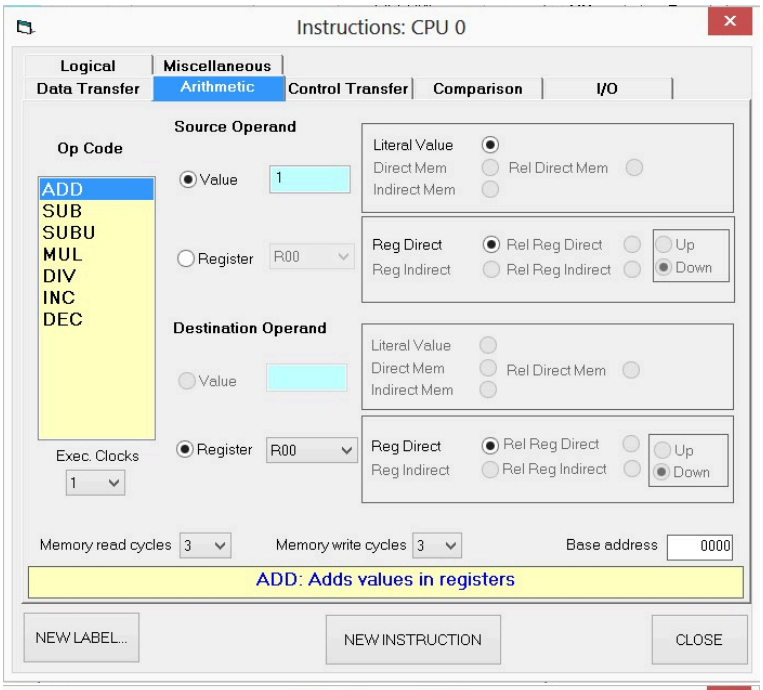


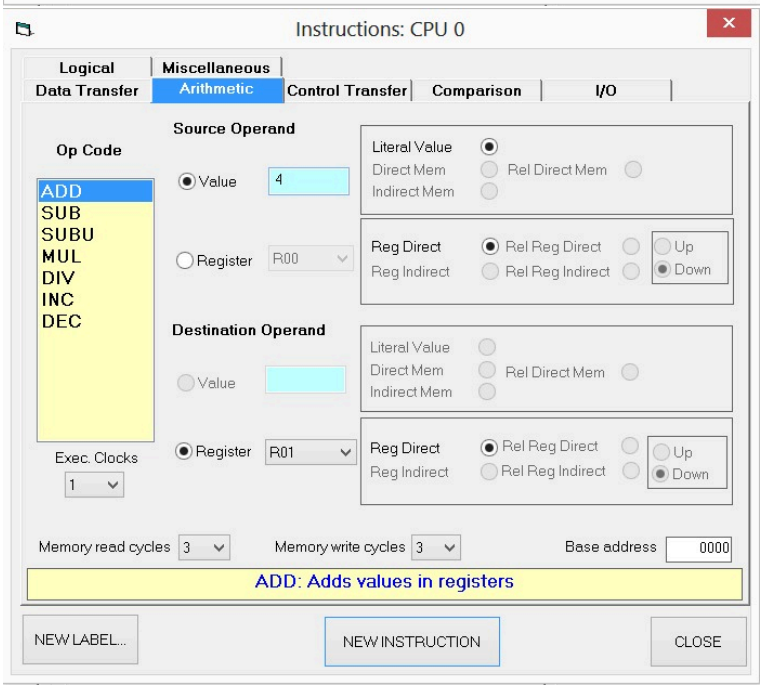
- Tampilan Register Set.



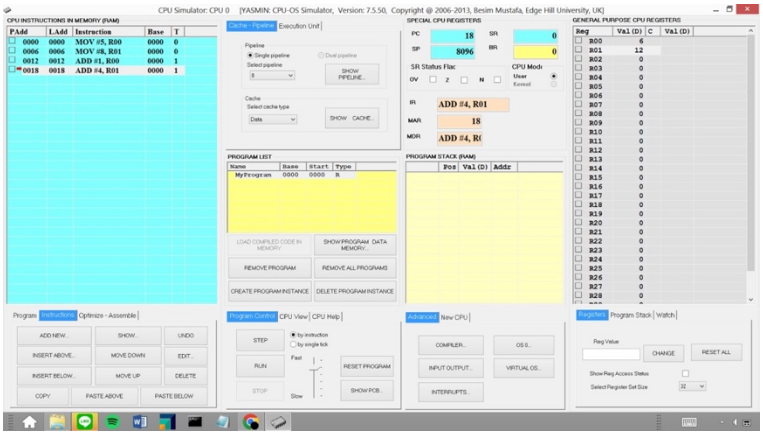
b. Aritmatika

- instruksi yang menambahkan (add) isi R00 dan R01



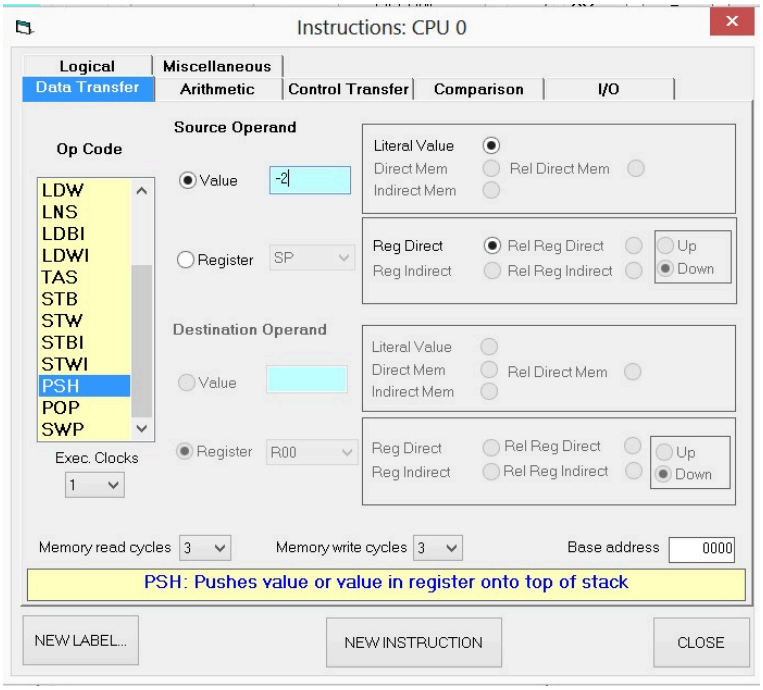


- hasil penjumlahan berada di Register Set

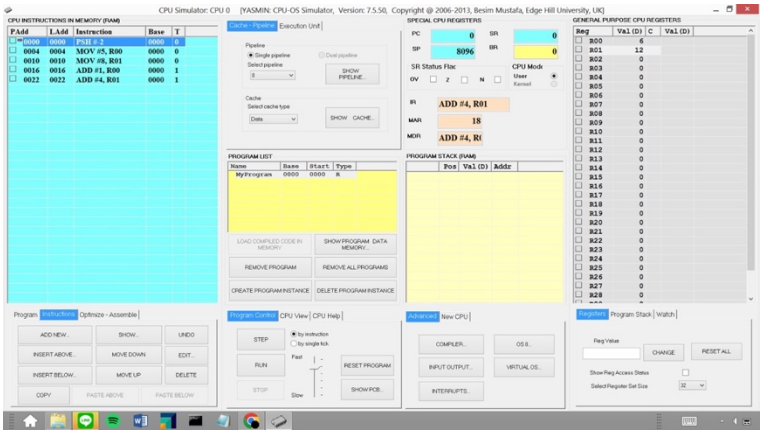


c. Stack Pointer (SP)

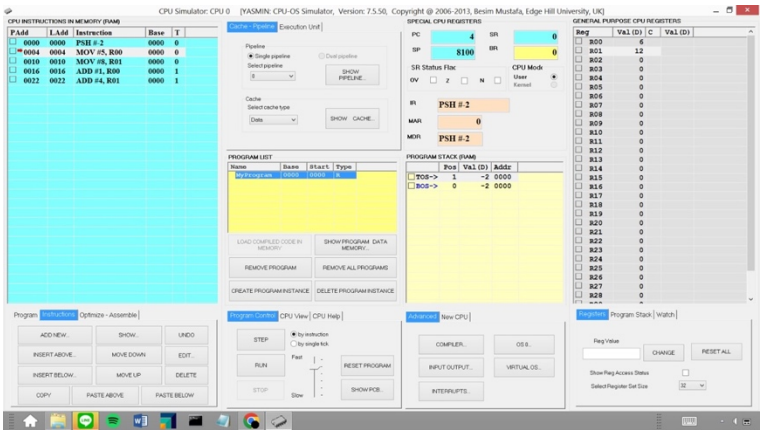
- Instruksi untuk menaruh (push) angka -2 pada stack teratas



- Tampilan Register Set.

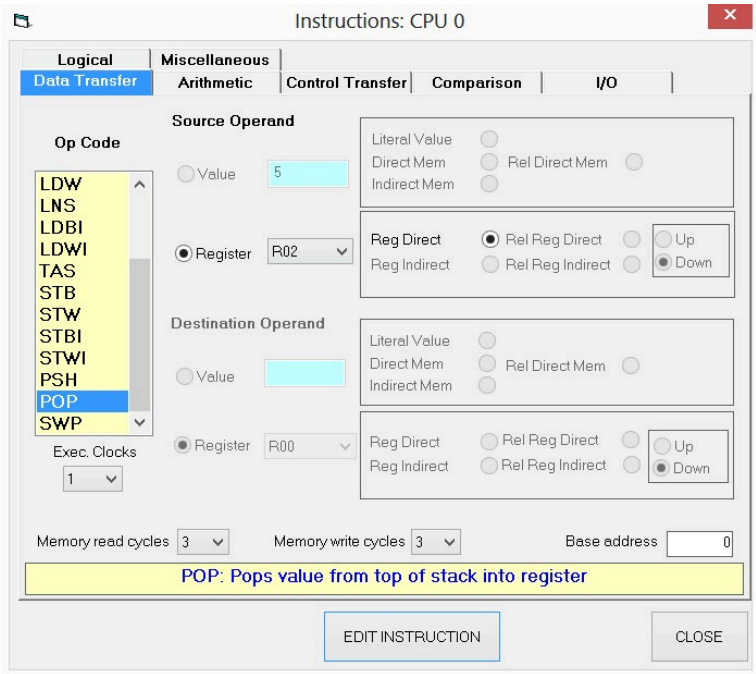


d. Pembanding

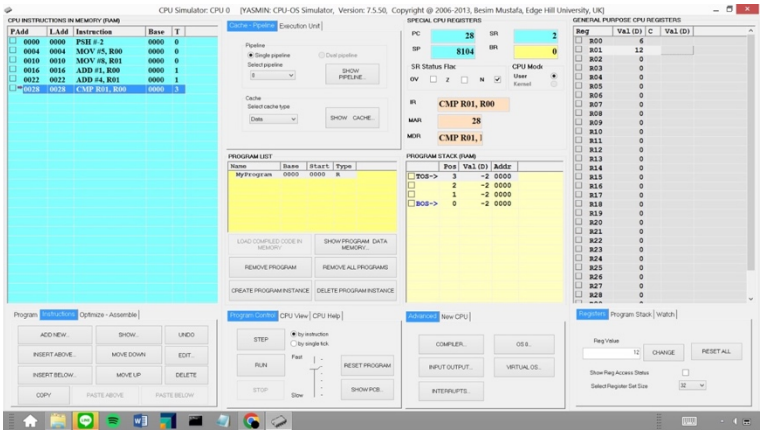


e. Stack Pointer (SP)

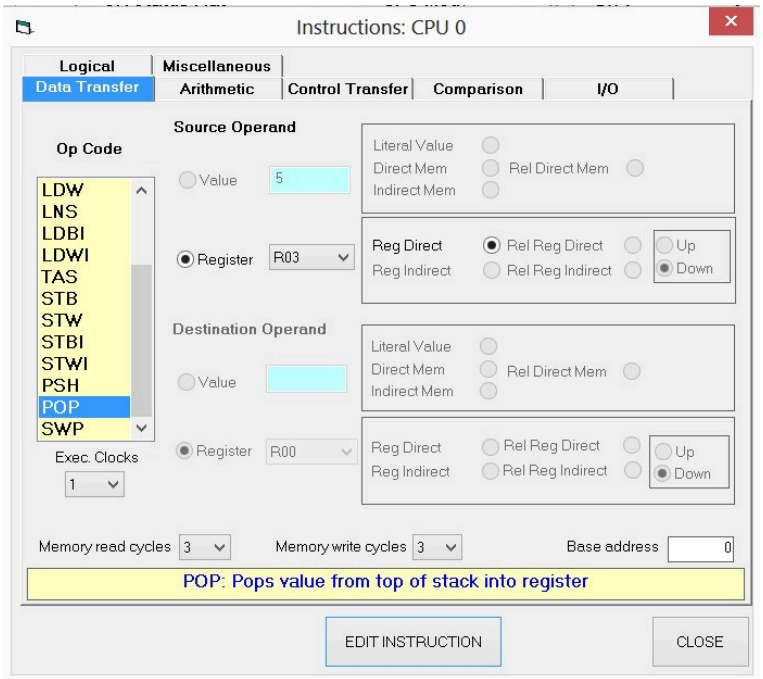
- Instruksi untuk mengambil (pop) nilai teratas dari program stack ke register R02.



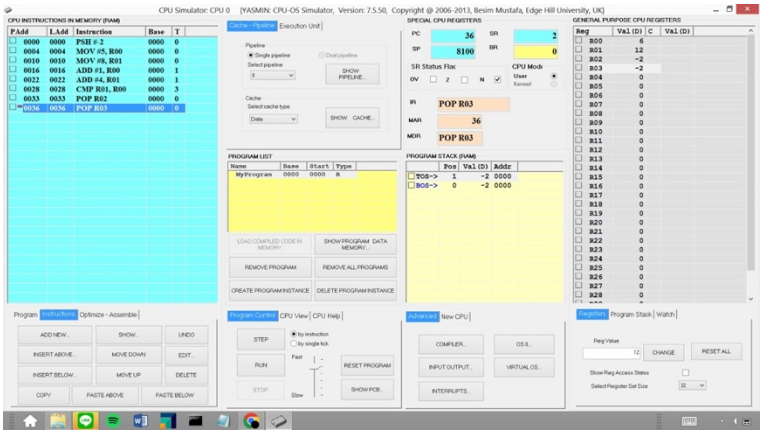
- Tampilan Register Set.



- Instruksi untuk mengambil (pop) nilai teratas dari program stack ke register R03.



- Tampilan Register Set.



PERTEMUAN 10

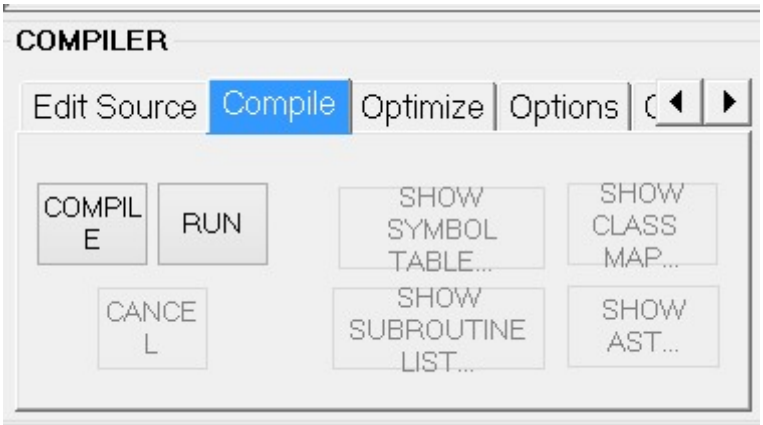
Menampilkan daftar proses/ thread dan pohon proses yang menggambarkan hubungan antar proses induk/anak :

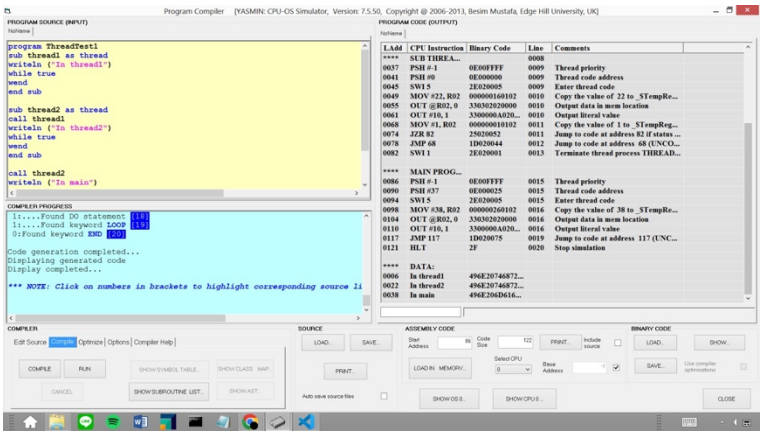
1. Kode

```
program ThreadTest1
sub thread1 as thread
writeln ("In thread1")
while true
wend
end sub

sub thread2 as thread
call thread1
writeln ("In thread2")
while true
wend
end sub

call thread2
```

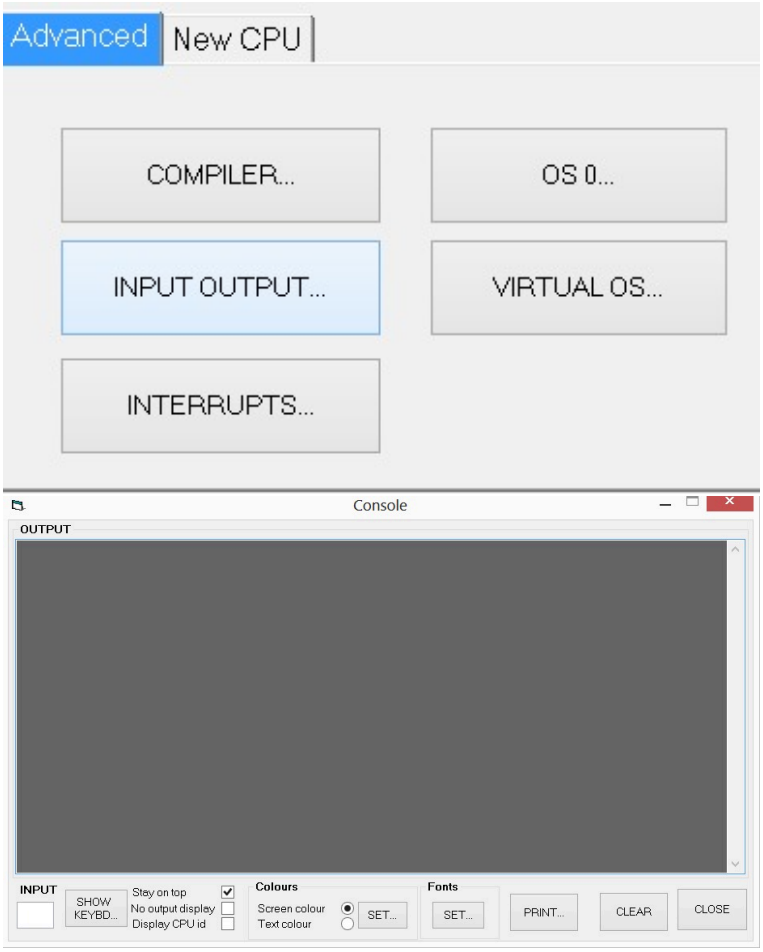
```
call thread2
writeln ("In main")

do
loop
end
```

2. Compile dengan menekan tombol compile di kotak compiler pada tab compile

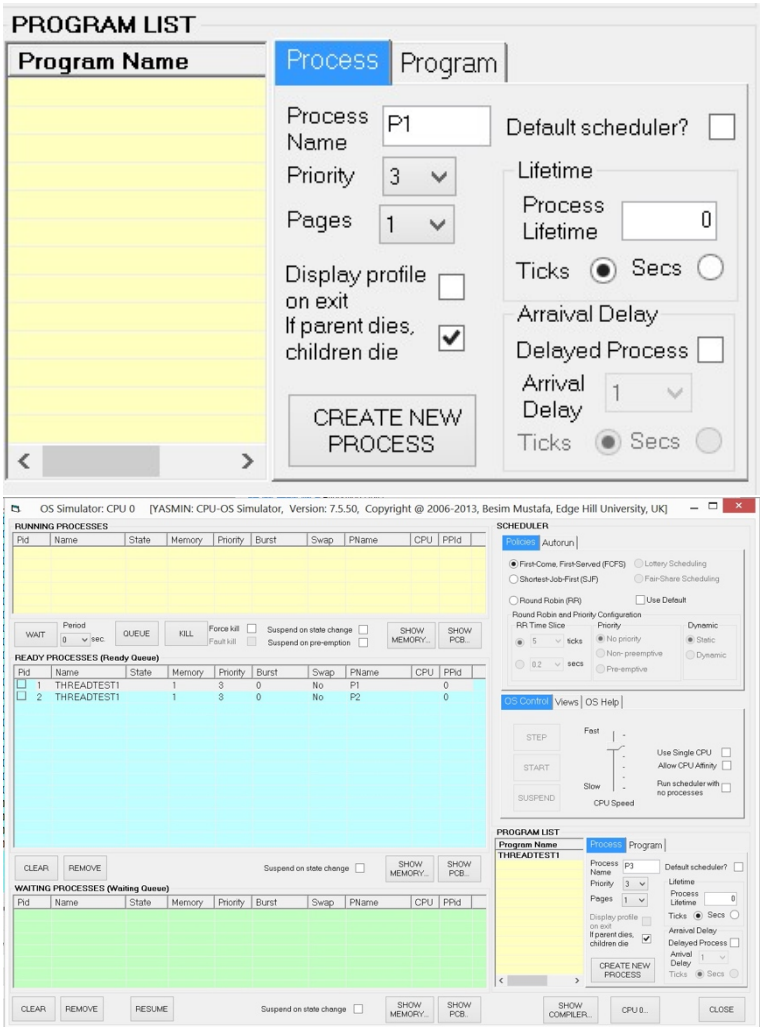




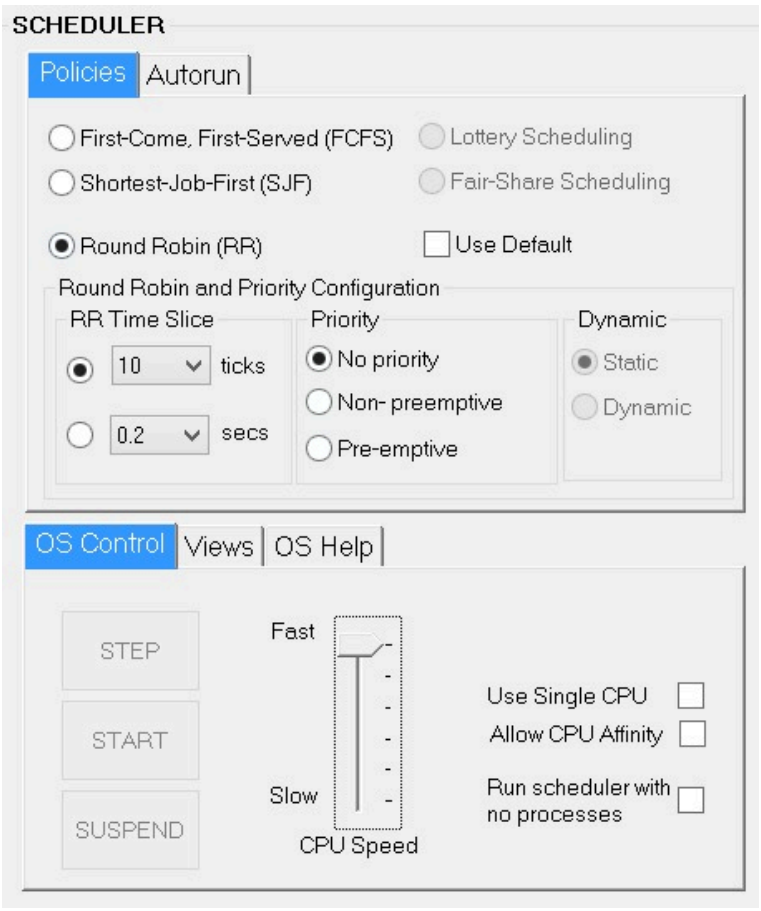
3. Klik tombol input/output... Aktifkan stay on top



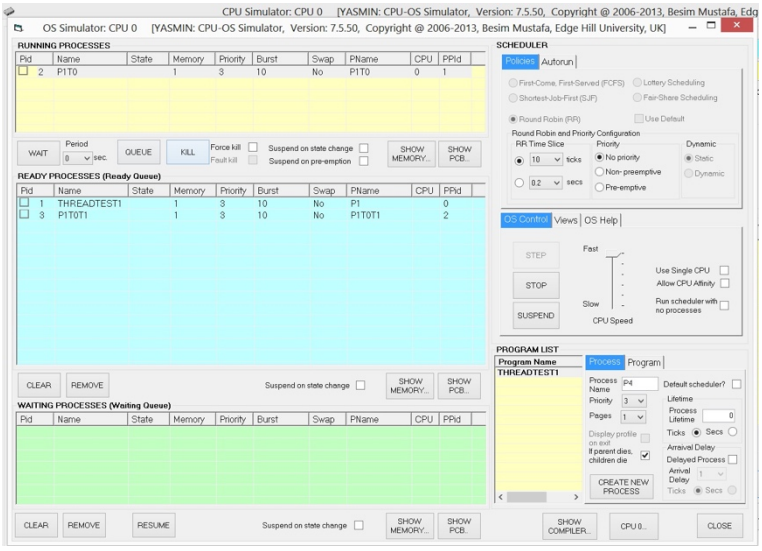
4. Buat satu proses dengan create new process.



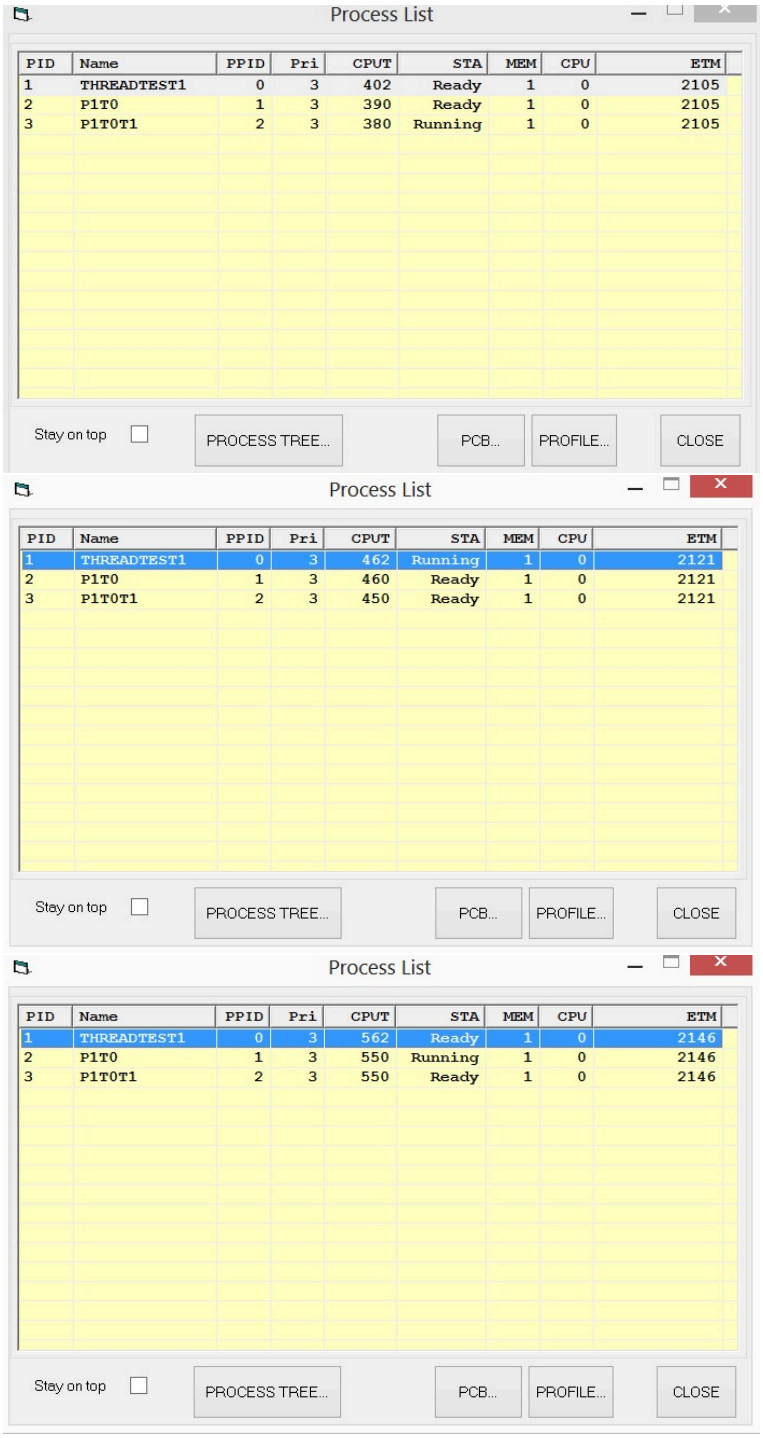
5. Jenis penjadwalan adalah round robin, 10 ticks dengan kecepatan simulasi maksimum.

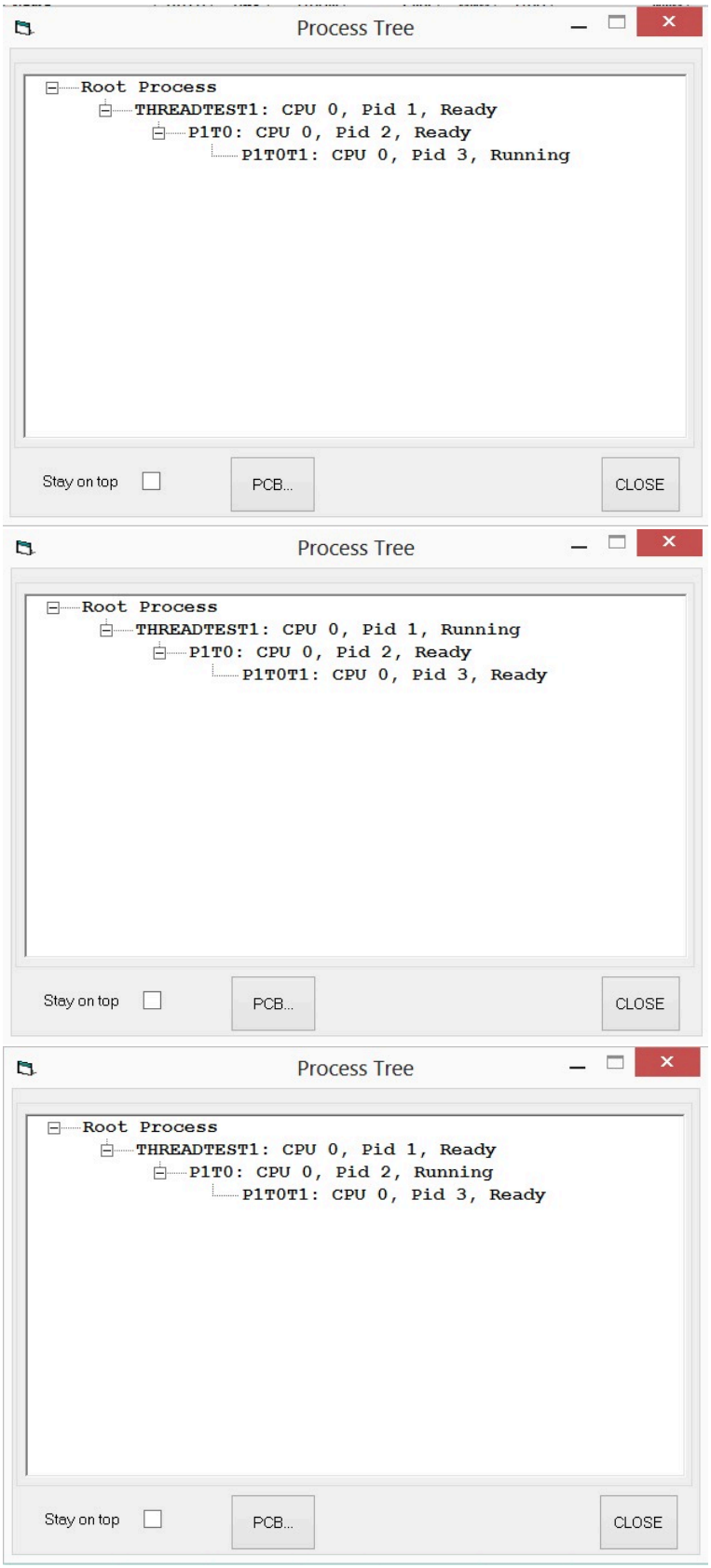


6. Tampilan pada jendela console.



7. Jumlah proses yang dibuat
8. Proses nya yang
9. Thread nya yang
10. Klik tombol VIEW PROCESS LIST. Kemudian klik PROCESS TREE..


Process List
PID Name PPID Pri CPUT STA MEM CPU ETM
1 THREADTEST1 0 3 402 Ready 1 0 2105
2 P1T0 1 3 390 Ready 1 0 2105
3 P1T0T1 2 3 380 Running 1 0 2105
Stay on top
PROCESS TREE...
PCB...
PROFILE...
CLOSE
Process List
PID Name PPID Pri CPUT STA MEM CPU ETM
1 THREADTEST1 0 3 462 Running 1 0 2121
2 P1T0 1 3 460 Ready 1 0 2121
3 P1T0T1 2 3 450 Ready 1 0 2121
Stay on top
PROCESS TREE...
PCB...
PROFILE...
CLOSE
Process List
PID Name PPID Pri CPUT STA MEM CPU ETM
1 THREADTEST1 0 3 562 Ready 1 0 2146
2 P1T0 1 3 550 Running 1 0 2146
3 P1T0T1 2 3 550 Ready 1 0 2146
Stay on top
PROCESS TREE...
PCB...
PROFILE...
CLOSE



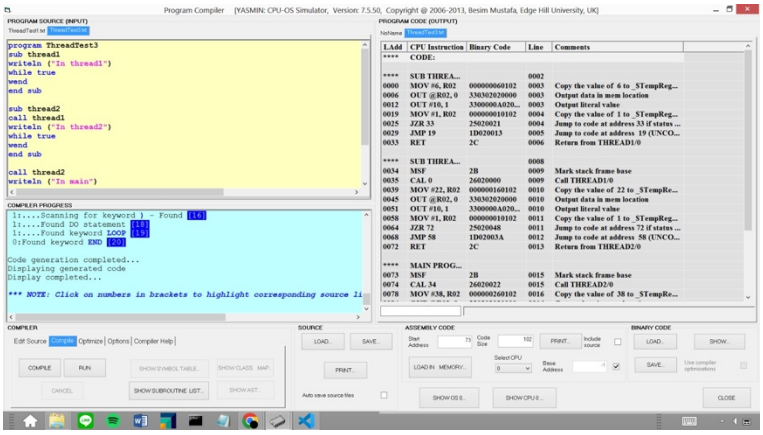
11. Identifikasi proses induk dan proses turunan!
12. tombol KILL untuk menghentikan proses!

Memodifikasi kode sumber untuk membuat versi tanpa thread dan membandingkannya dengan versi yang menggunakan thread.

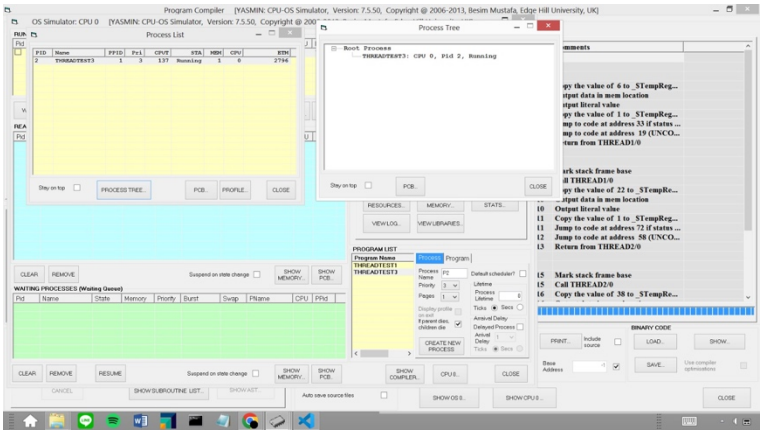
1. Modifikasi source code ThreadTest1 dengan menghapus instance “as thread” pada deklarasi subroutine. Ganti nama program menjadi ThreadTest3.

```
program ThreadTest3
sub thread1
writeln ("In thread1")
while true
wend
end sub

sub thread2
call thread1
writeln ("In thread2")
while true
wend
end sub

call thread2
writeln ("In main")
```

2. Compile code hasil modifikasi tersebut.



3. Buat proses baru lagi, kemudian jalankan di OS simulator.
4. Amati tampilan pada jendela console, apa perbedaan proses tanpa thread dan proses menggunakan thread?



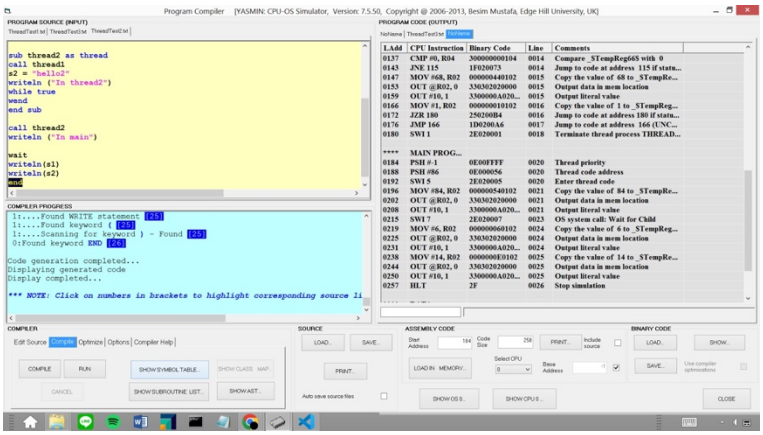
Memahami bagaimana thread-thread berbagi sumber daya dari proses induk.

1. Kode

```
program ThreadTest2
var s1 string(6)
var s2 string(6)

sub thread1 as thread
s1 = "hello1"
writeln ("In thread1")
while true
wend
end sub

sub thread2 as thread
call thread1
s2 = "hello2"
writeln ("In thread2")
while true
```

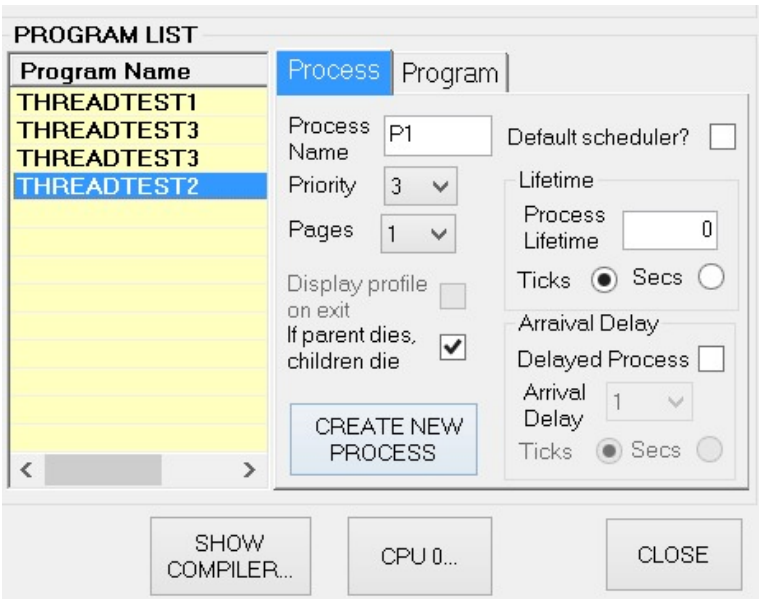
```
sub thread2 as thread
call thread1
s2 = "hello2"
writeln ("In thread2")
while true
wend
end sub

call thread2
writeln ("In main")

wait
writeln(s1)
writeln(s2)
end
```

2. Compile program ThreadTest2. Klik tombol SYMBOL TABLE pada jendela compiler. Amati informasi tentang variabel s1 dan s2.

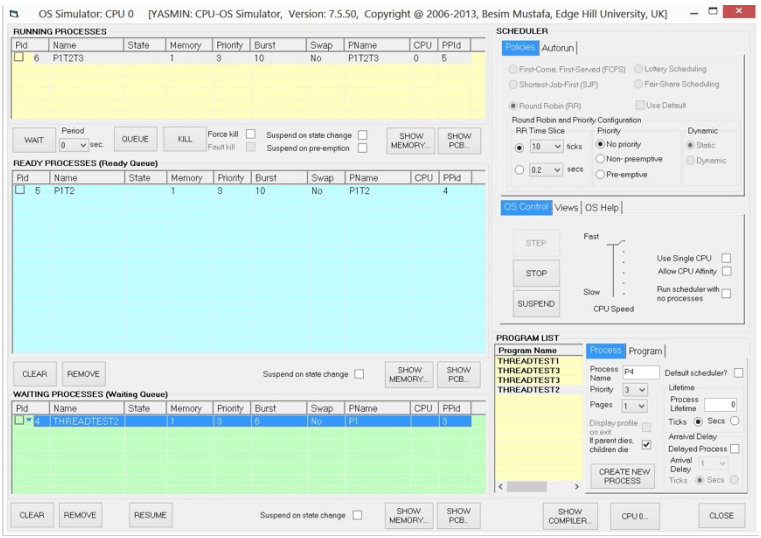


3. Melalui CPU Simulator, tampilkan jendela konsol dengan klik tombol INPUT/OUTPUT... aktifkan STAY ON TOP.

4. Beralihlah ke jendela OS Simulator, buat satu proses dengan CREATE NEW PROCESS.

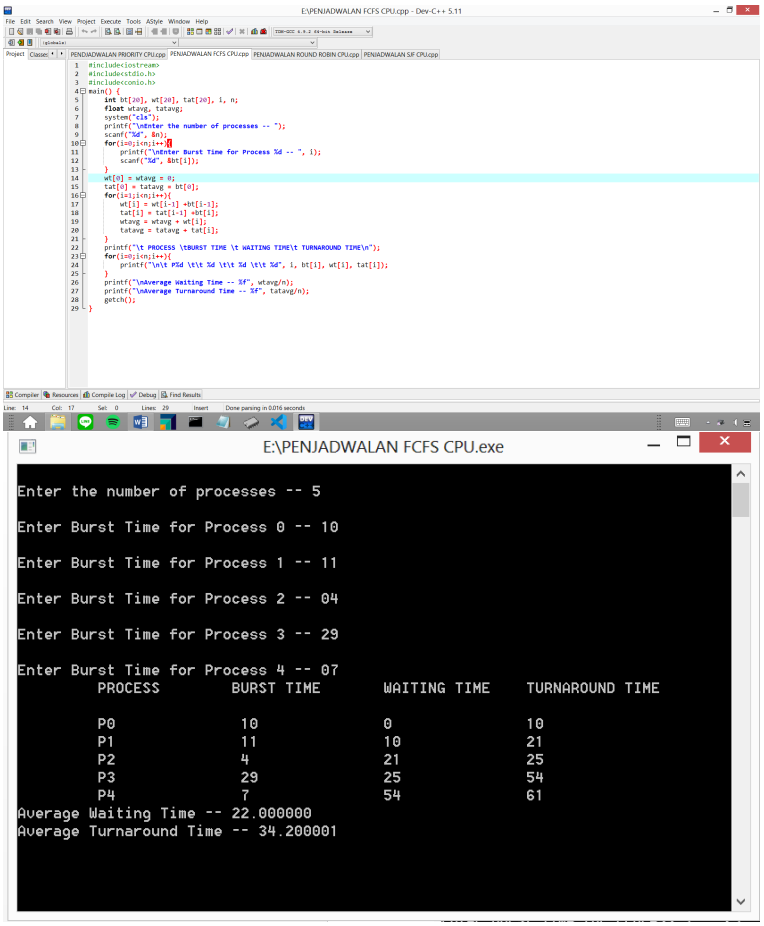


5. Pastikan jenis penjadwalan adalah ROUND ROBIN, kecepatan simulasi maksimum.



PERTEMUAN 11

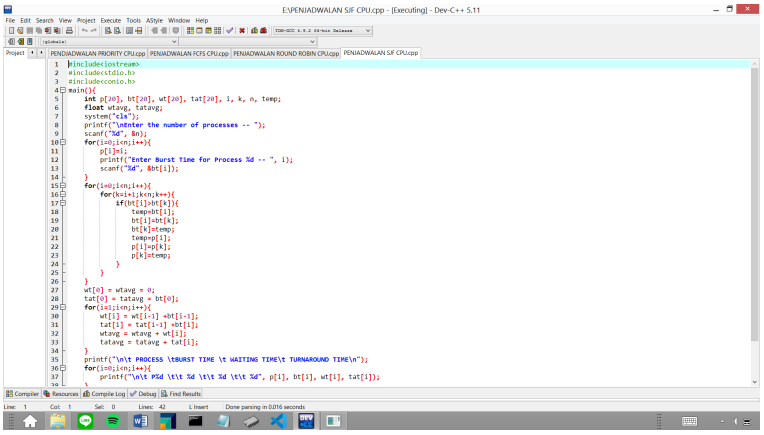
A. Algoritma penjadwalan FCFS CPU



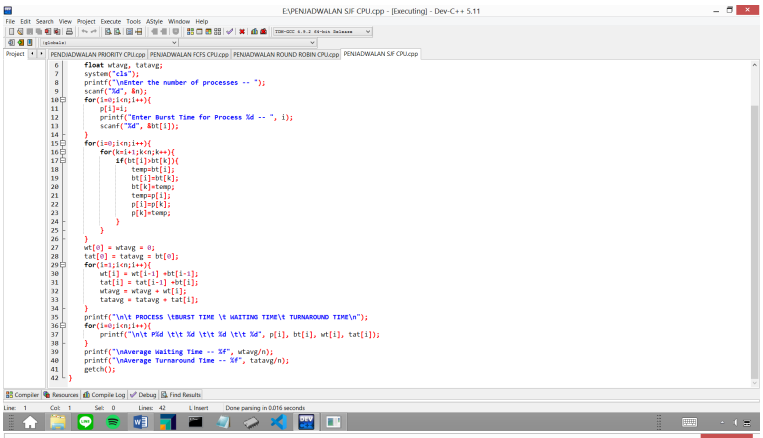
```
#include<iostream>
#include<stdio.h>
#include<conio.h>
main() {
    int bt[20], wt[20], tat[20], i, n;
    float wtavg, tatavg;
    system("cls");
    printf("\nEnter the number of processes -- ");
    scanf("%d", &n);
    for(i=0;i<n;i++){
        printf("\nEnter Burst Time for Process %d -- ", i);
        scanf("%d", &bt[i]);
    }
    wt[0] = wtavg = 0;
    tat[0] = tatavg = bt[0];
    for(i=1;i<n;i++){
        wt[i] = wt[i-1] +bt[i-1];
        tat[i] = tat[i-1] +bt[i];
        wtavg = wtavg + wt[i];
        tatavg = tatavg + tat[i];
    }
    printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
    for(i=0;i<n;i++){
        printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);
    }
    printf("\nAverage Waiting Time -- %f", wtavg/n);
    printf("\nAverage Turnaround Time -- %f", tatavg/n);
    getch();
}
```

```
E:\PENJADWALAN FCFS CPU.exe

Enter the number of processes -- 5

Enter Burst Time for Process 0 -- 10

Enter Burst Time for Process 1 -- 11

Enter Burst Time for Process 2 -- 04

Enter Burst Time for Process 3 -- 29

Enter Burst Time for Process 4 -- 07
         PROCESS        BURST TIME       WAITING TIME     TURNAROUND TIME

         P0             10               0                10
         P1             11               10               21
         P2             4                21               25
         P3             29               25               54
         P4             7                54               61
Average Waiting Time -- 22.000000
Average Turnaround Time -- 34.200001
```

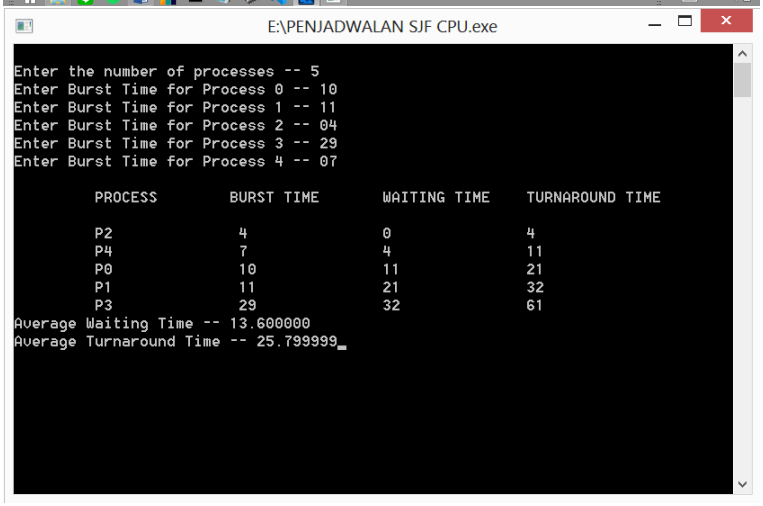
B. Algoritma penjadwalan SJF CPU



```
#include<iostream>
#include<stdio.h>
#include<conio.h>
main(){
    int p[20], bt[20], wt[20], tat[20], i, k, n, temp;
    float wtavg, tatavg;
    system("cls");
    printf("\nEnter the number of processes -- ");
    scanf("%d", &n);
    for(i=0;i<n;i++){
        p[i]=i;
        printf("Enter Burst Time for Process %d -- ", i);
        scanf("%d", &bt[i]);
    }
    for(i=0;i<n;i++){
        for(k=i+1;k<n;k++){
            if(bt[i]>bt[k]){
                temp=bt[i];
                bt[i]=bt[k];
                bt[k]=temp;
                temp=p[i];
                p[i]=p[k];
                p[k]=temp;
            }
        }
    }
    wt[0] = wtavg = 0;
    tat[0] = tatavg = bt[0];
    for(i=1;i<n;i++){
        wt[i] = wt[i-1] +bt[i-1];
        tat[i] = tat[i-1] +bt[i];
        wtavg = wtavg + wt[i];
        tatavg = tatavg + tat[i];
    }
    printf("\n\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
    for(i=0;i<n;i++){
        printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]);
    }
```

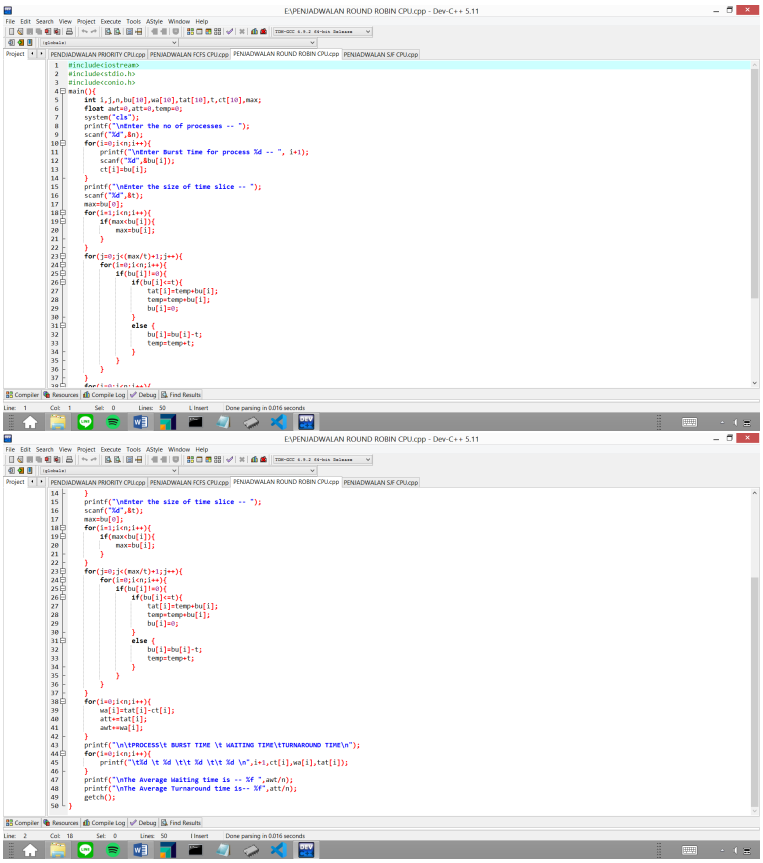


```
float wtavg, tatavg;
system("cls");
printf("\nEnter the number of processes -- ");
scanf("%d", &n);
for(i=0;i<n;i++){
    p[i]=i;
    printf("Enter Burst Time for Process %d -- ", i);
    scanf("%d", &bt[i]);
}
for(i=0;i<n;i++){
    for(k=i+1;k<n;k++){
        if(bt[i]>bt[k]){
            temp=bt[i];
            bt[i]=bt[k];
            bt[k]=temp;
            temp=p[i];
            p[i]=p[k];
            p[k]=temp;
        }
    }
}
wt[0] = wtavg = 0;
tat[0] = tatavg = bt[0];
for(i=1;i<n;i++){
    wt[i] = wt[i-1] +bt[i-1];
    tat[i] = tat[i-1] +bt[i];
    wtavg = wtavg + wt[i];
    tatavg = tatavg + tat[i];
}
printf("\n\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
for(i=0;i<n;i++){
    printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]);
}
printf("\nAverage Waiting Time -- %f", wtavg/n);
printf("\nAverage Turnaround Time -- %f", tatavg/n);
getch();
}
```

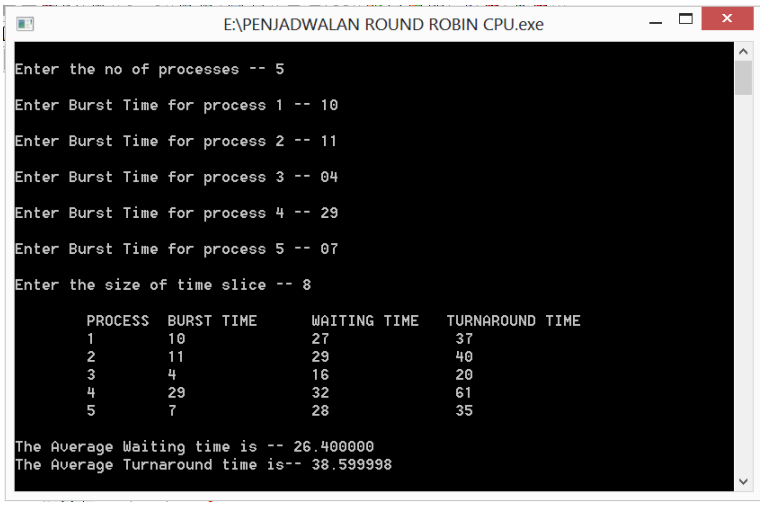


```
E:\PENJADWALAN SJF CPU.exe

Enter the number of processes -- 5
Enter Burst Time for Process 0 -- 10
Enter Burst Time for Process 1 -- 11
Enter Burst Time for Process 2 -- 04
Enter Burst Time for Process 3 -- 29
Enter Burst Time for Process 4 -- 07

        PROCESS         BURST TIME      WAITING TIME    TURNAROUND TIME

        P2              4               0               4
        P4              7               4               11
        P0              10              11              21
        P1              11              21              32
        P3              29              32              61
Average Waiting Time -- 13.600000
Average Turnaround Time -- 25.799999
```

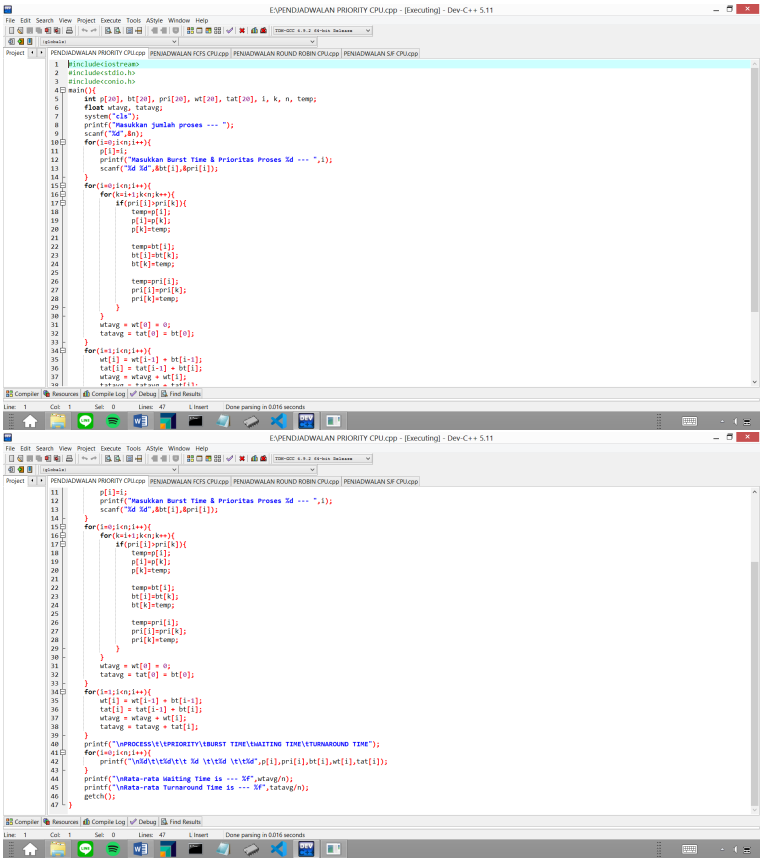
C. Algoritma penjadwalan ROUND ROBIN CPU

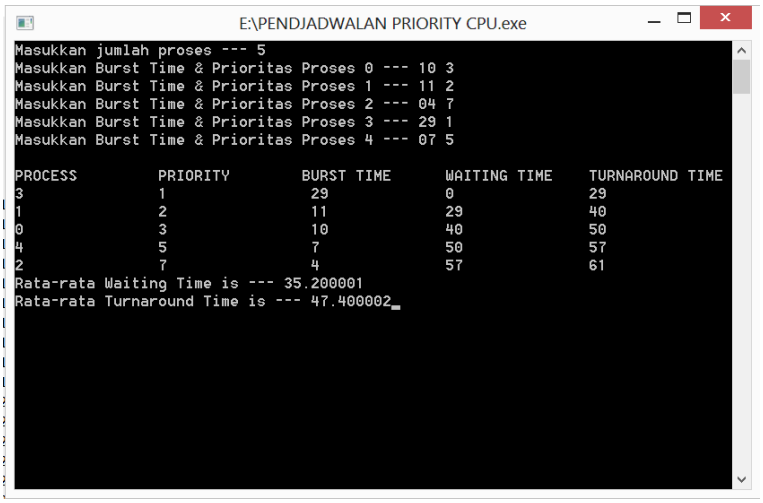


```
#include<iostream>
#include<stdio.h>
#include<conio.h>
main(){
    int i,j,n,bu[10],wa[10],tat[10],t,ct[10],max;
    float awt=0,att=0,temp=0;
    system("cls");
    printf("\nEnter the no of processes -- ");
    scanf("%d",&n);
    for(i=0;i<n;i++){
        printf("\nEnter Burst Time for process %d -- ", i+1);
        scanf("%d",&bu[i]);
        ct[i]=bu[i];
    }
    printf("\nEnter the size of time slice -- ");
    scanf("%d",&t);
    max=bu[0];
    for(i=1;i<n;i++){
        if(max<bu[i]){
            max=bu[i];
        }
    }
    for(j=0;j<(max/t)+1;j++){
        for(i=0;i<n;i++){
            if(bu[i]!=0){
                if(bu[i]<=t){
                    tat[i]=temp+bu[i];
                    temp=temp+bu[i];
                    bu[i]=0;
                }
                else {
                    bu[i]=bu[i]-t;
                    temp=temp+t;
                }
            }
        }
    }
    for(i=0;i<n;i++){
        wa[i]=tat[i]-ct[i];
        att+=tat[i];
        awt+=wa[i];
    }
    printf("\n\tPROCESS\t BURST TIME \t WAITING TIME\tTURNAROUND TIME\n");
    for(i=0;i<n;i++){
        printf("\tP%d \t %d \t\t %d \t\t %d \n",i+1,ct[i],wa[i],tat[i]);
    }
    printf("\nThe Average Waiting time is -- %f ",awt/n);
    printf("\nThe Average Turnaround time is-- %f",att/n);
    getch();
}
```



D. Algoritma penjadwalan PRIORITY CPU





Kesimpulannya adalah algoritma SJF merupakan algoritma dengan waktu tunggu dan waktu putar nya paling cepat dan algoritma PRIORITY adalah yang paling lambat (dalam percobaan diatas). Akan berbeda hasil jika algoritma ROUND ROBIN dan PRIORITY diubah size time of slice dan urutan prioritas nya.

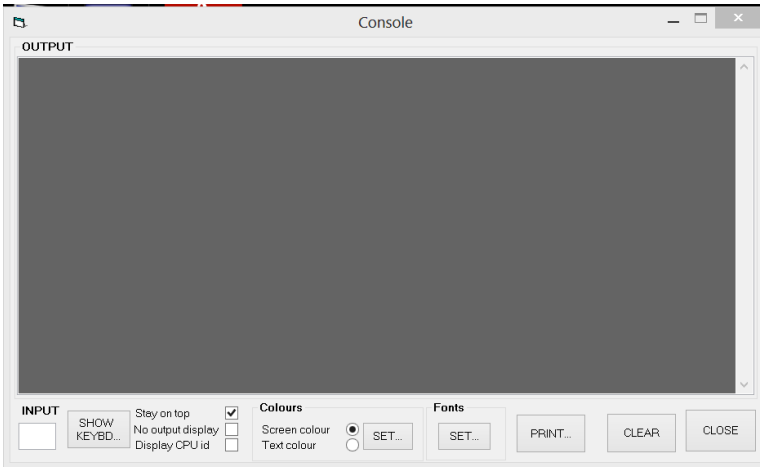
PERTEMUAN 12

1. Proses diwaktu yang bersamaan
   - Kode program

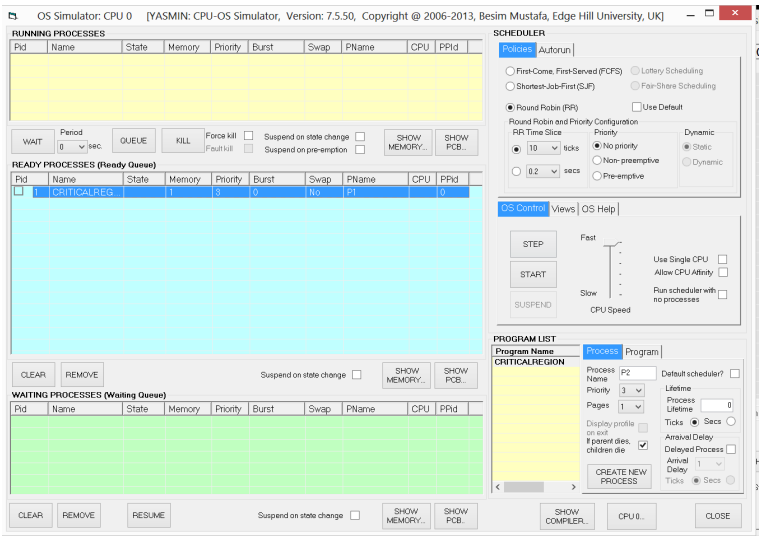
```
program CriticalRegion
     var g integer
     sub thread1 as thread
          writeln("In thread1")
          g = 0
          for n = 1 to 20
               g = g + 1
          next
          writeln("thrad1 g = ", g)
          writeln("Exiting thread1")
     end sub
     sub thread2 as thread
          writeln("In thread2")
          g = 0
          for n = 1 to 12
```

```
     sub thread2 as thread
          writeln("In thread2")
          g = 0
          for n = 1 to 12
               g = g + 1
          next
          writeln("thrad2 g = ", g)
          writeln("Exiting thread2")
     end sub
     writeln("In main")
     call thread1
     call thread2
     wait
     writeln("Exiting main")
end
```

   - Mengaktifkan STAY ON TOP



   - Di OS simulator buat proses denngan ROUND ROBIN, 10 ticks, kecepatan maksimum.



- Output di jendela console

```
In main
Exiting main
In main
In thread1
In thread2
thrad2 g =  18
Exiting thread2
thrad1 g =  25
Exiting thread1
Exiting main
```

2. Proses dengan synchronise
   - Kode program

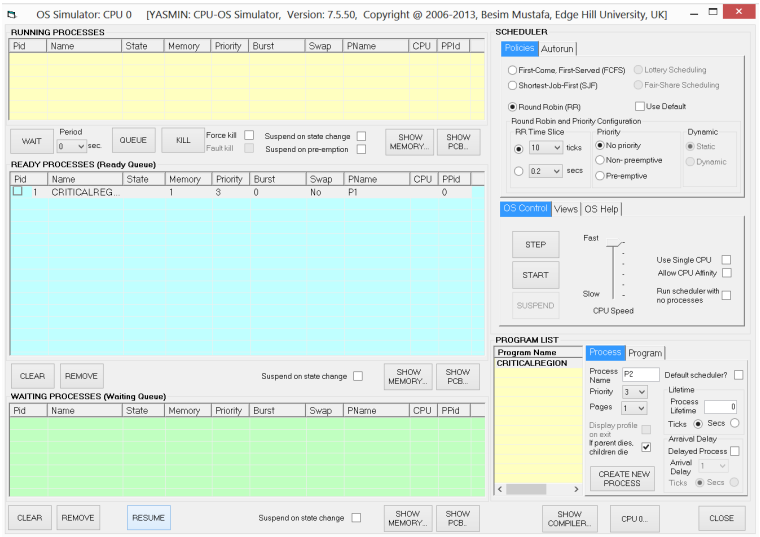
```
program CriticalRegion
     var g integer
     sub thread1 as thread synchronise
          writeln("In thread1")
          g = 0
          for n = 1 to 20
               g = g + 1
          next
          writeln("thrad1 g = ", g)
          writeln("Exiting thread1")
     end sub
     sub thread2 as thread synchronise
          writeln("In thread2")
          g = 0
          for n = 1 to 12
```

```
          writeln("In thread2")
          g = 0
          for n = 1 to 12
               g = g + 1
          next
          writeln("thrad2 g = ", g)
          writeln("Exiting thread2")
     end sub
     writeln("In main")
     call thread1
     call thread2
     wait
     writeln("Exiting main")
end
```

- Baris di program code (output) yang menerjemahkan synchronise.

LAdd	CPU Instruction	Binary Code	Line	Comments
****	SUB THREA...		0003	
0000	TAS 0, R01	090200000101	0003	Test and set
0006	CMP #0, R01	300000000101	0003	Compare _STempReg94$ with 0
0012	JEQ 24	1E020018	0003	Jump to code at address 24 if the sta...
0016	PSH #0	0E000000	0003	Push thread start address
0020	SWI 10	2E02000A	0003	OS system call: Blocked Wait

- Di OS simulator buat proses denngan ROUND ROBIN, 10 ticks, kecepatan maksimum.



- Output di jendela console

```
In main
Exiting main
In main
Exiting main
In main
In thread1
thrad1 g =  20
Exiting thread1
In thread2
thrad2 g =  12
Exiting thread2
Exiting main
```

3. Proses dengan enter dan leave
   - Kode program

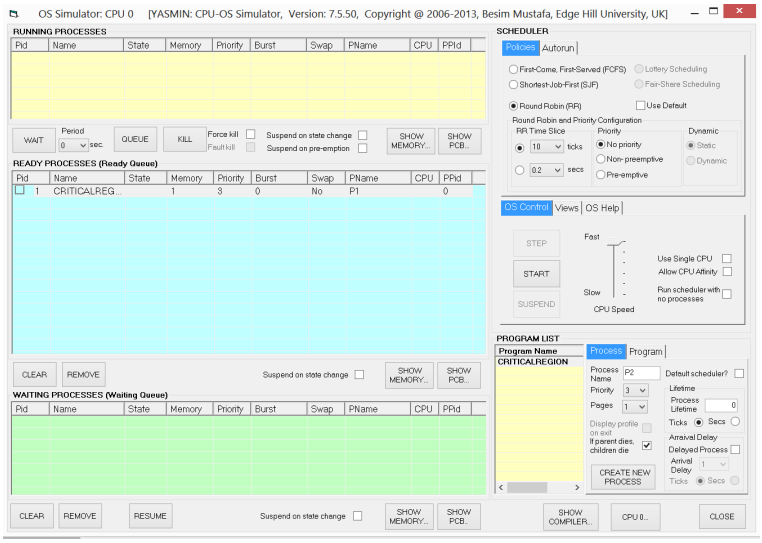
```
program CriticalRegion
     var g integer
     sub thread1 as thread
          writeln("In thread1")
          enter
          g = 0
          for n = 1 to 20
               g = g + 1
          next
          writeln("thrad1 g = ", g)
          leave
          writeln("Exiting thread1")
     end sub
     sub thread2 as thread
          writeln("In thread2")
```

```
            enter
            g = 0
            for n = 1 to 12
                 g = g + 1
            next
            writeln("thrad2 g = ", g)
            leave
            writeln("Exiting thread2")
      end sub
      writeln("In main")
      call thread1
      call thread2
      wait
      writeln("Exiting main")
end
```

- Baris di program code (output) yang menerjemahkan enter dan leave.

0019	SWI 18	2E020012	0005	OS system call: Enter critical region
0130	SWI 19	2E020013	0011	OS system call: Exit critical region

- Di OS simulator buat proses denngan ROUND ROBIN, 10 ticks, kecepatan maksimum.



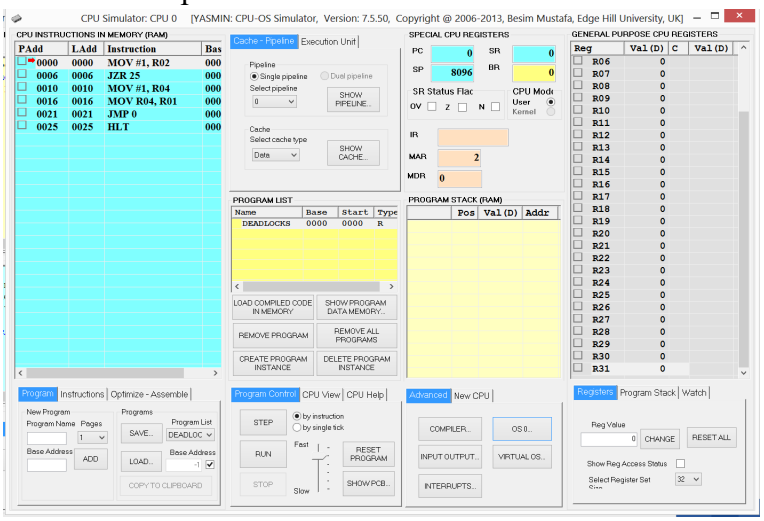
- Output di jendela console

```
In main
Exiting main
In main
In thread1
In thread2
thrad1 g =  20
Exiting thread1
thrad2 g =  12
Exiting thread2
Exiting main
```

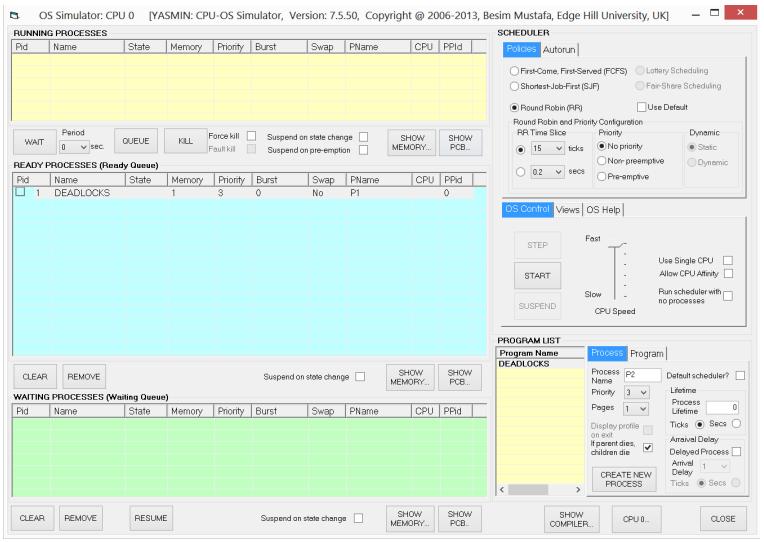
4. Deadlock
   - Kode program

```
Program Deadlocks
      while true
            n = 1
      wend
end
```

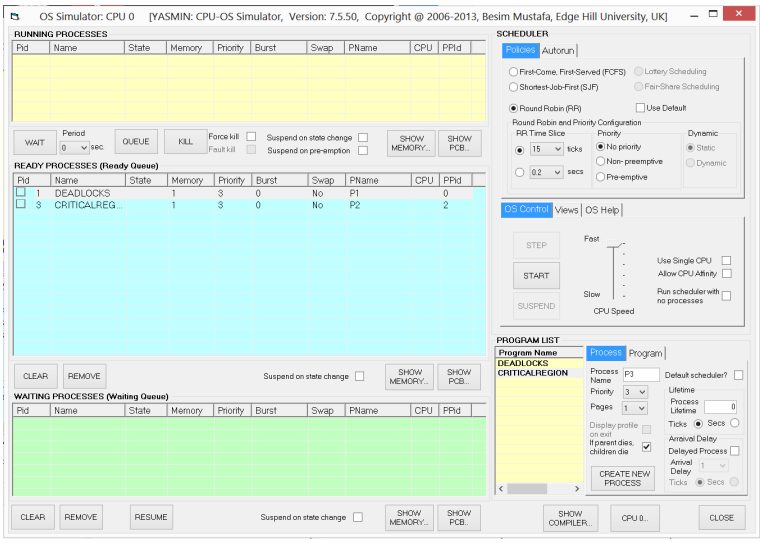
- Hasil compile



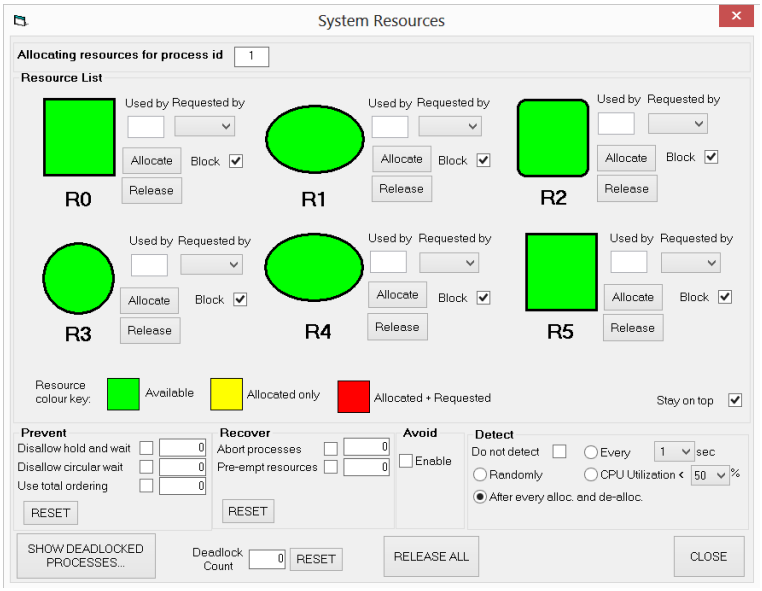
- Di OS simulator buat proses denngan ROUND ROBIN, 10 ticks, kecepatan maksimum.



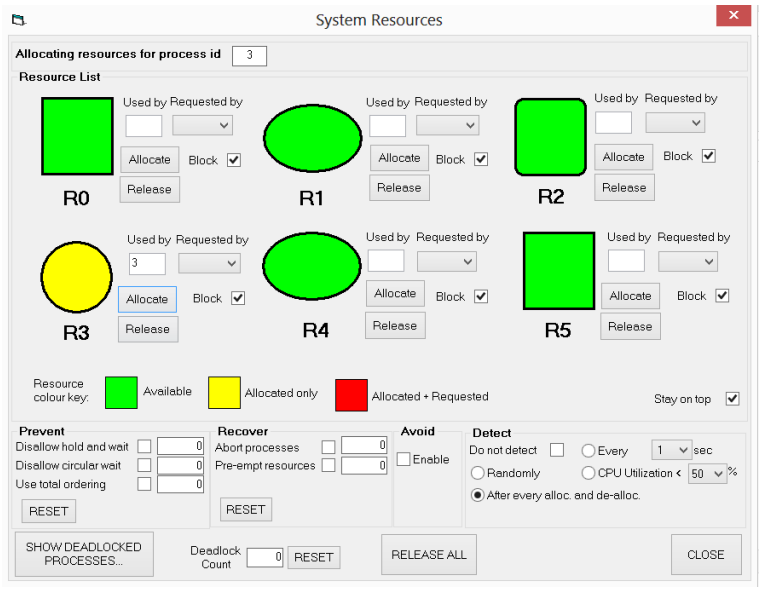
- Menambahkan program lain pada antrian



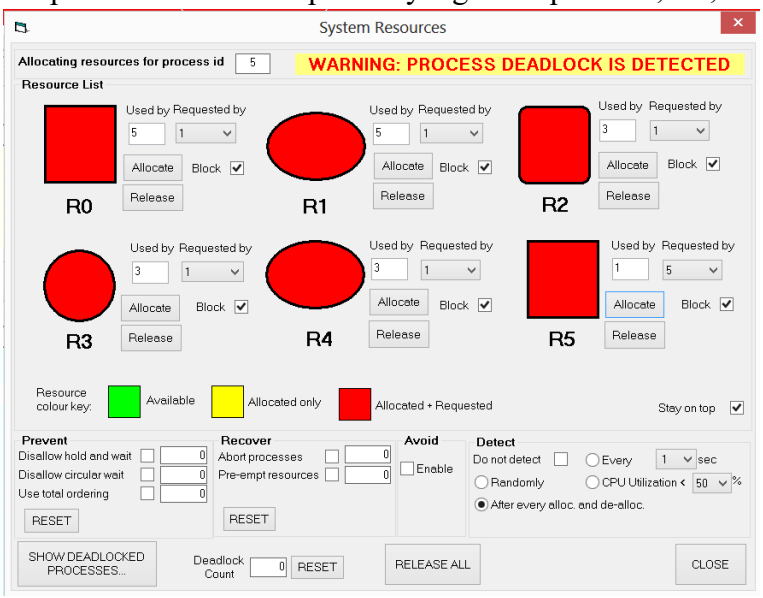
- Tampilan awal pada jendela VIEW RESOURCES



- Tampilan pada jendela VIEW RESOURCES setelah mengalokasikan proses id 3



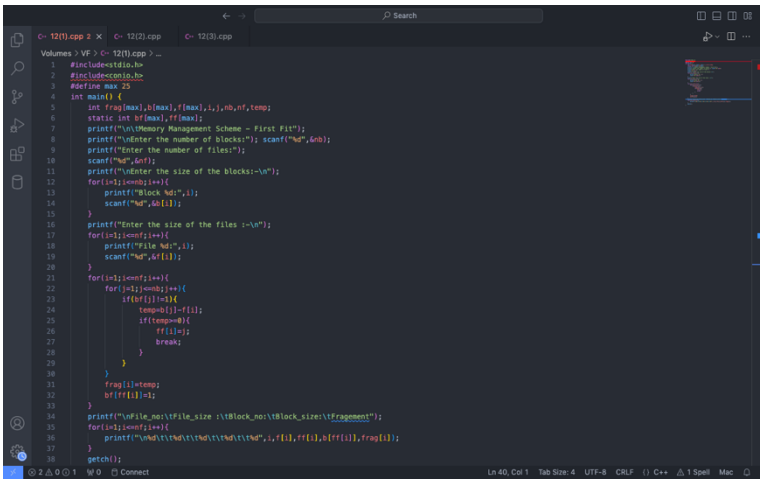
- Tampilan pada jendela VIEW RESOURCES setelah mengalokasikan proses id 3 dan id 1 pada R yang sama pada R0, R1, R2, R3, R4, R5.



PERTEMUAN 13

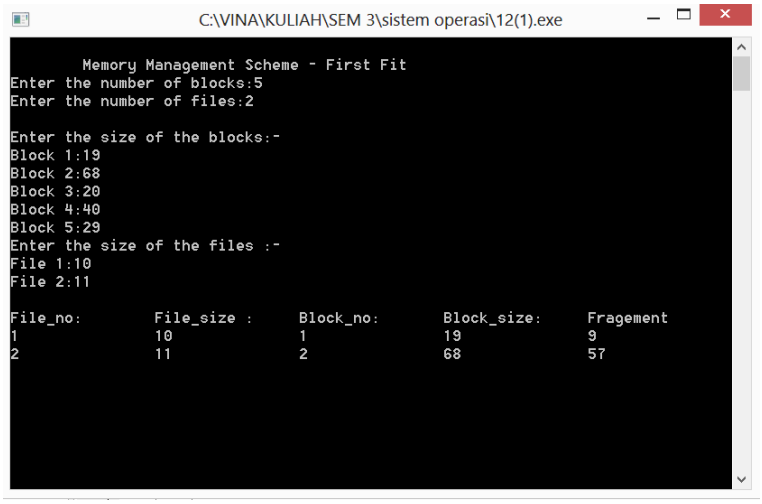
1. First Fit

   Input :



```
#include<stdio.h>
#include<conio.h>
#define max 25
int main() {
    int frag[max],b[max],f[max],i,j,nb,nf,temp;
    static int bf[max],ff[max];
    printf("\n\tMemory Management Scheme - First Fit");
    printf("\nEnter the number of blocks:"); scanf("%d",&nb);
    printf("Enter the number of files:");
    scanf("%d",&nf);
    printf("\nEnter the size of the blocks:-\n");
    for(i=1;i<=nb;i++){
        printf("Block %d:",i);
        scanf("%d",&b[i]);
    }
    printf("Enter the size of the files :-\n");
    for(i=1;i<=nf;i++){
        printf("File %d:",i);
        scanf("%d",&f[i]);
    }
    for(i=1;i<=nf;i++){
        for(j=1;j<=nb;j++){
            if(bf[j]!=1){
                temp=b[j]-f[i];
                if(temp>=0){
                    ff[i]=j;
                    break;
                }
            }
        }
        frag[i]=temp;
        bf[ff[i]]=1;
    }
    printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
    for(i=1;i<=nf;i++){
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
    }
    getch();
```

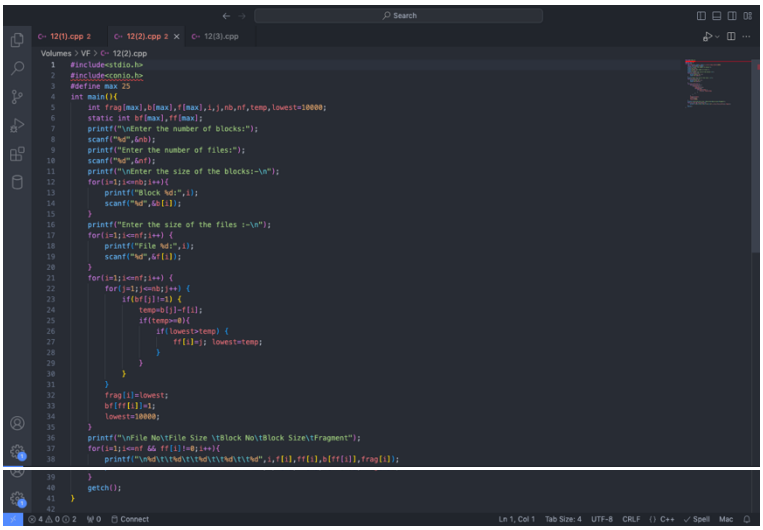
   Output :



```
C:\VINA\KULIAH\SEM 3\sistem operasi\12(1).exe

        Memory Management Scheme - First Fit
Enter the number of blocks:5
Enter the number of files:2

Enter the size of the blocks:-
Block 1:19
Block 2:68
Block 3:20
Block 4:40
Block 5:29
Enter the size of the files :-
File 1:10
File 2:11

File_no:        File_size :     Block_no:       Block_size:     Fragement
1               10              1               19              9
2               11              2               68              57
```

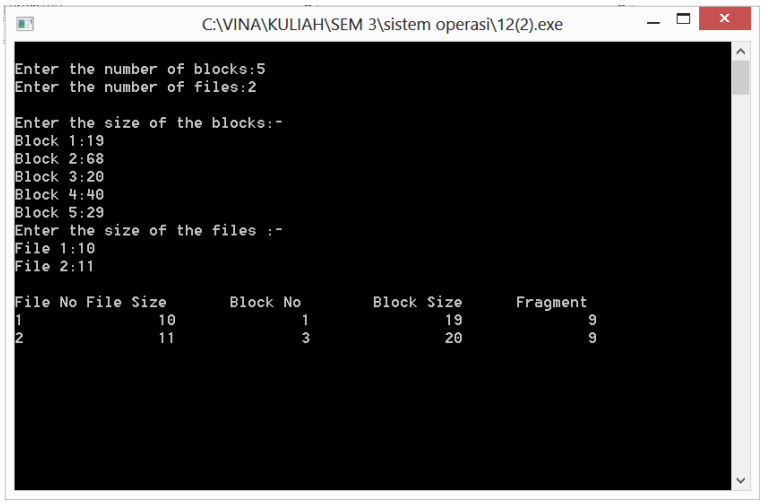
2. Best Fit

   Input :



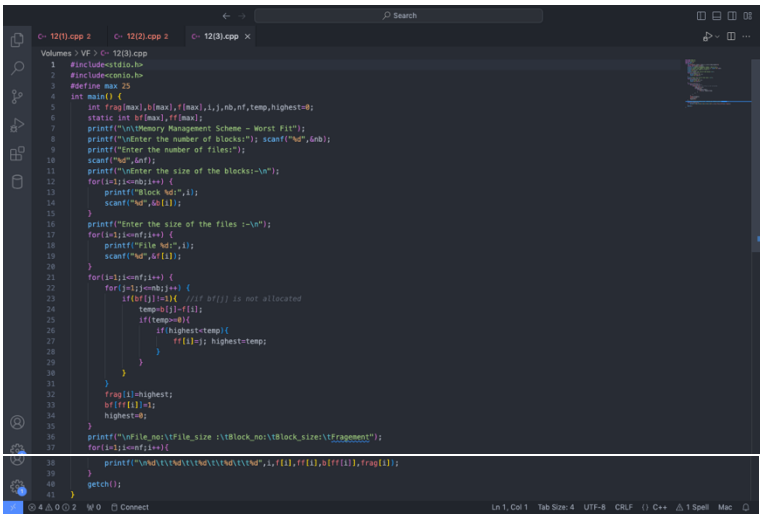
```
#include<stdio.h>
#include<conio.h>
#define max 25
int main(){
    int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
    static int bf[max],ff[max];
    printf("\nEnter the number of blocks:");
    scanf("%d",&nb);
    printf("Enter the number of files:");
    scanf("%d",&nf);
    printf("\nEnter the size of the blocks:-\n");
    for(i=1;i<=nb;i++){
        printf("Block %d:",i);
        scanf("%d",&b[i]);
    }
    printf("Enter the size of the files :-\n");
    for(i=1;i<=nf;i++) {
        printf("File %d:",i);
        scanf("%d",&f[i]);
    }
    for(i=1;i<=nf;i++) {
        for(j=1;j<=nb;j++) {
            if(bf[j]!=1) {
                temp=b[j]-f[i];
                if(temp>=0){
                    if(lowest>temp) {
                        ff[i]=j; lowest=temp;
                    }
                }
            }
        }
        frag[i]=lowest;
        bf[ff[i]]=1;
        lowest=10000;
    }
    printf("\nFile No\tFile Size \tBlock No\tBlock Size\tFragment");
    for(i=1;i<=nf && ff[i]!=0;i++){
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
    }
    getch();
}
```

   Output :

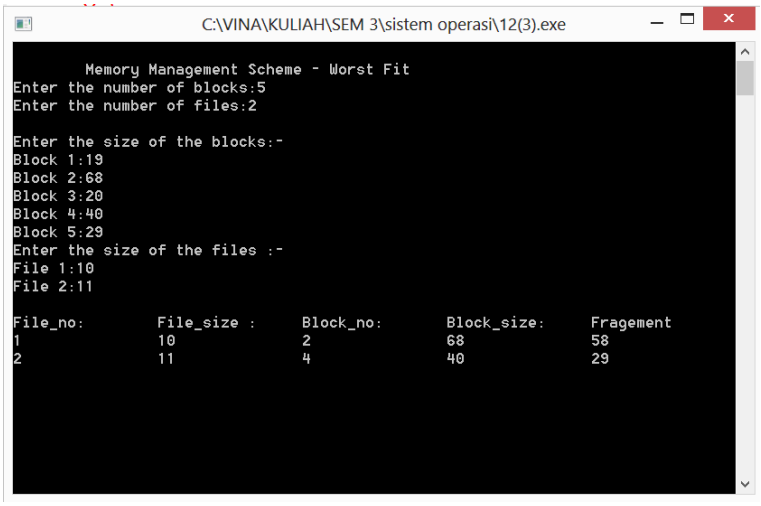


3. Worst Fit

Input :



Output :



Kesimpulan :

- First Fit memulai pencarian dari awal daftar memori dan menggunakan blok pertama yang terdaftar.

- Best Fit mencari blok yang pas (kecil) untuk digunakan.
- Worst Fit mencari blok paling besar dan menggunakanya.