



Experiment No.6
Implement process scheduling algorithms FCFS and SJF using CPU-OS Simulator.
Date of Performance:
Date of Submission:



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

---

**Aim:** To study and implement process scheduling algorithms FCFS and SJF using CPU-OS Simulator.

**Objective:** Its main objective is to increase CPU utilization and hence the throughput of the system by keeping the CPU as busy as possible.

### **Theory:**

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms.

These algorithms are either non-preemptive or preemptive. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

### **First Come First Serve (FCFS)**

Jobs are executed on first come, first serve basis. It is a non-preemptive, pre-emptive scheduling algorithm. Easy to understand and implement. Its implementation is based on FIFO queue. Poor in performance as average wait time is high.

### **Shortest Job First (SJF)**

This is also known as shortest job first, or SJF. This is a non-preemptive, pre-emptive scheduling algorithm. Best approach to minimize waiting time. Easy to implement in Batch systems where required CPU time is known in advance. Impossible to implement in interactive systems where required CPU time is not known. The processor should know in advance how much time process will take.

Installation steps of CPU-OS simulator:

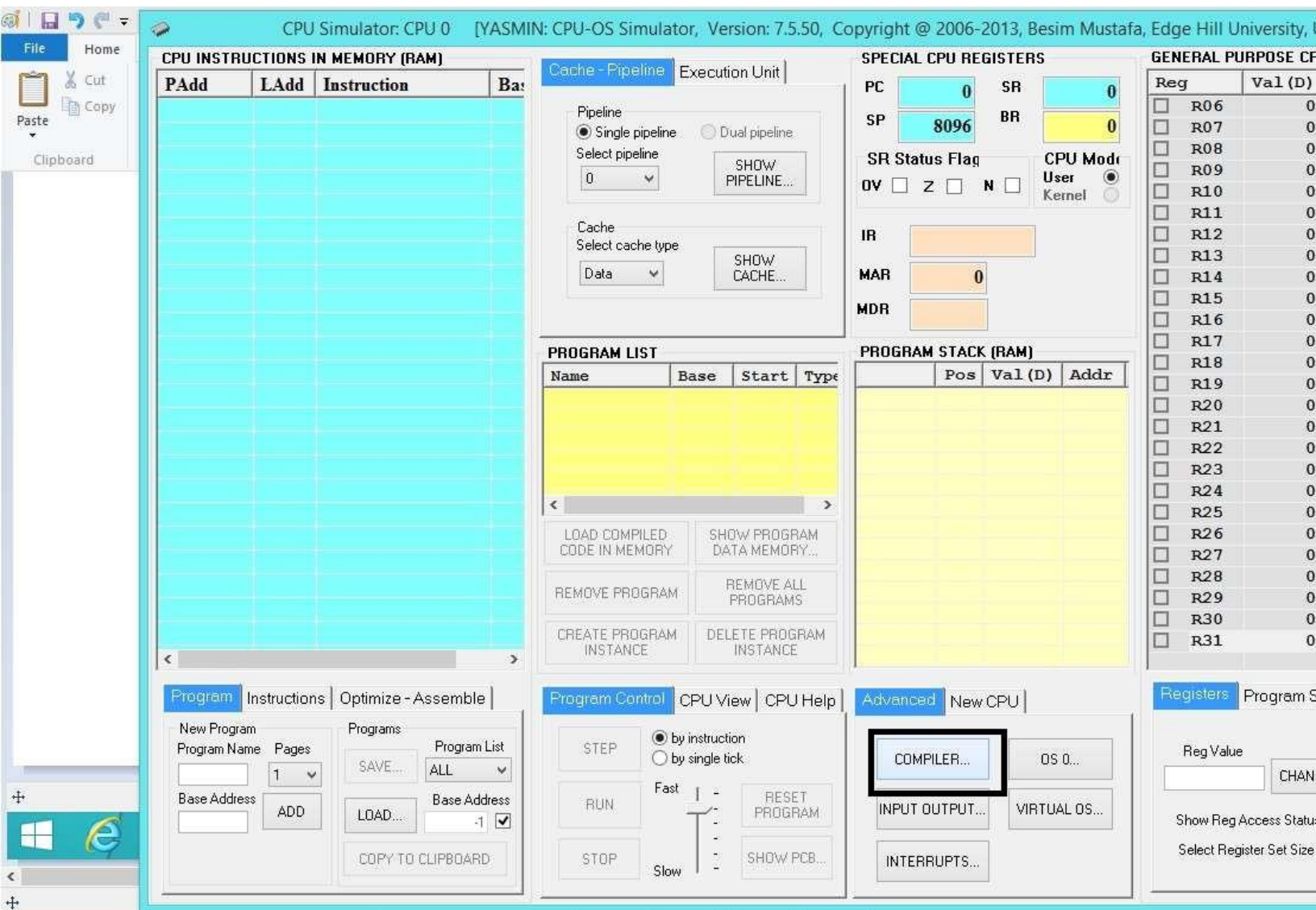
- Go to <http://www.teach-sim.com/>. Select downloads and get the latest version of the simulator. Install the simulator using the downloaded executable file. It will install the CPU-OS Simulator on your system.
- Or you can install it from [cpu-os-simulator.software.informer.com](http://cpu-os-simulator.software.informer.com) website.



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

- Open the compiler window by selecting the **COMPILER** button in the current window. You should now be looking at the compiler window.



- In the compiler window, enter the following source code in the compiler's source editor window (under PROGRAM SOURCE frame title):

PROGRAM LoopTest

i = 0

for n = 0 to 40

i = i + 1



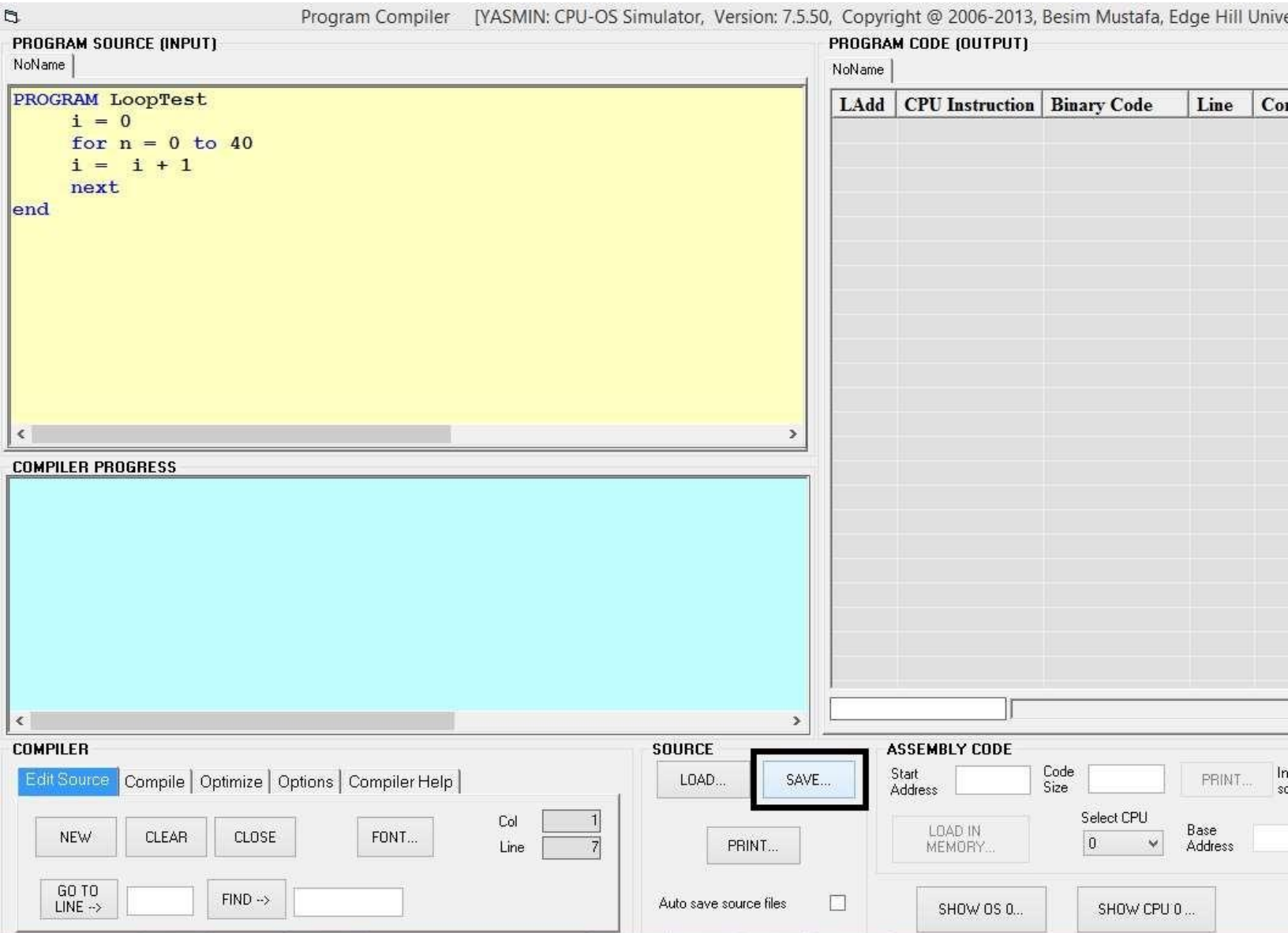
# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

next

end

- Save your source code.



- Click on the **COMPILE** button. You should see the code created on the right in **PROGRAM CODE** view.



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

Program Compiler [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright © 2006-2013, Besim Mustafa, Edge Hill U]

**PROGRAM SOURCE (INPUT)**  
pro2.txt

```
PROGRAM LoopTest
i = 0
for n = 0 to 40
i = i + 1
next
end
```

**PROGRAM CODE (OUTPUT)**  
NoName

LAdd	CPU Instruction	Binary Code	Line
------	-----------------	-------------	------

**COMPILER PROGRESS**

**COMPILER**  
Edit Source **Compile** Optimize Options Compiler Help

**COMPILE** RUN

CANCEL

SHOW SYMBOL TABLE... SHOW CLASS MAP...  
SHOW SUBROUTINE LIST... SHOW AST...

**SOURCE**  
LOAD... SAVE...  
PRINT...  
Auto save source files ☐

**ASSEMBLY CODE**  
Start Address Code Size PRINT...  
LOAD IN MEMORY... Select CPU Base Address  
0  
SHOW OS 0... SHOW CPU 0...

- Click on the button **SHOW** in **BINARY CODE** view. You should now see the **Hexadecimal Code for LOOPTEST** window





# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

Program Compiler [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright @ 2006-2013, Besim Mustafa, Edge Hill]

**PROGRAM SOURCE (INPUT)**  
pro2.txt  

```
PROGRAM LoopTest
  i = 0
  for n = 0 to 40
    i = i + 1
  next
end
```

**COMPILER PROGRESS**  
2:.....Found Assignment statement [4]  
1:.....Found keyword NEXT [5]  
0:Found keyword END [6]  
Code generation completed...  
Displaying generated code  
Display completed...  
\*\*\* NOTE: Click on numbers in brackets to highlight correspondi

**PROGRAM CODE (OUTPUT)**  
NoName  

LAdd	CPU Instruction	Binary Code	Line
****	CODE:		
****	MAIN PROG...		
0000	MOV #0, R03	000000000103	0002
0006	MOV R03, R01	0001030101	0002
0011	MOV #0, R04	000000000104	0003
0017	MOV R04, R02	0001040102	0003
0022	MOV #40, R04	000000280104	0003
0028	CMP R04, R02	3001040102	0003
0033	JGT 68	20020044	0003
0037	MOV R01, R03	0001010103	0004
0042	ADD #1, R03	110000010103	0004
0048	MOV R03, R05	0001030105	0004
0053	MOV R05, R01	0001050101	0004
0058	ADD #1, R02	110000010102	0005
0064	JMP 28	1D02001C	0005
0068	HLT	2F	0006
****	DATA:		

**COMPILER**  
Edit Source **Compile** Optimize Options Compiler Help  
[COMPILE] [RUN] [SHOW SYMBOL TABLE....] [SHOW CLASS MAP...]  
[CANCEL] [SHOW SUBROUTINE LIST...] [SHOW AST...]

**SOURCE**  
[LOAD...] [SAVE...]  
[PRINT...]  
Auto save source files ☐

**ASSEMBLY CODE**  
Start Address: 0 Code Size: 69 [PRINT...]  
[LOAD IN MEMORY...]  
Select CPU: 0 Base Address:  
[SHOW OS 0...] [SHOW CPU 0...]



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

- Now, this code needs to be loaded in memory so that the CPU can execute it. To do this,

Program Compiler [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright © 2006-2013, Besim Mustafa, Edge Hill]

**PROGRAM SOURCE (INPUT)**  
pro2.txt

```
PROGRAM LoopTest
i = 0
for n = 0 to 40
i = i + 1
next
end
```

**COMPILER PROGRESS**

```
2:.....Found Assignment statement [4]
1:....Found keyword NEXT [5]
0:Found keyword END [6]

Code generation completed...
Displaying generated code
Display completed...

*** NOTE: Click on numbers in brackets to highlight correspondi
```

**PROGRAM CODE (OUTPUT)**  
NoName

LAdd	CPU Instruction	Binary Code	Line
****	CODE:		
****	MAIN PROG...		
0000	MOV #0, R03	000000000103	0002
0006	MOV R03, R01	0001030101	0002
0011	MOV #0, R04	000000000104	0003
0017	MOV R04, R02	0001040102	0003
0022	MOV #40, R04	000000280104	0003
0028	CMP R04, R02	3001040102	0003
0033	JGT 68	20020044	0003
0037	MOV R01, R03	0001010103	0004
0042	ADD #1, R03	110000010103	0004
0048	MOV R03, R05	0001030105	0004
0053	MOV R05, R01	0001050101	0004
0058	ADD #1, R02	110000010102	0005
0064	JMP 28	1D02001C	0005
0068	HLT	2F	0006
****	DATA:		

**COMPILER**  
Edit Source **Compile** Optimize Options Compiler Help

COMPILE RUN SHOW SYMBOL TABLE... SHOW CLASS MAP... CANCEL SHOW SUBROUTINE LIST... SHOW AST...

**SOURCE**  
LOAD... SAVE... PRINT... Auto save source files ☐

**ASSEMBLY CODE**  
Start Address: 0 Code Size: 69 PRINT...  
LOAD IN MEMORY... Select CPU: 0 Base Address:  
SHOW OS 0... SHOW CPU 0...

click on the **LOAD IN MEMORY** button in the current window.



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

- You should now see the code loaded in memory ready to be executed. You are also back in the CPU simulator at this stage.

The screenshot displays the YASMIN CPU-OS Simulator interface. The main window is titled "CPU Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright © 2006-2013, Besim Mustafa, Edge Hill]".

**Program Source:** The left pane shows the source code for "pro2.txt":`PROGRAM Loc
i = 0
for n
i = i
next
end`

**CPU INSTRUCTIONS IN MEMORY (RAM):** The central pane displays a table of instructions loaded into memory:

PAdd	LAdd	Instruction	Base
0000	0000	MOV #0, R03	000
0006	0006	MOV R03, R01	000
0011	0011	MOV #0, R04	000
0017	0017	MOV R04, R02	000
0022	0022	MOV #40, R04	000
0028	0028	CMP R04, R02	000
0033	0033	JGT 68	000
0037	0037	MOV R01, R03	000
0042	0042	ADD #1, R03	000
0048	0048	MOV R03, R05	000
0053	0053	MOV R05, R01	000
0058	0058	ADD #1, R02	000
0064	0064	JMP 28	000
0068	0068	HLT	000

**Cache-Pipeline:** The "Cache-Pipeline" tab is active, showing options for "Single pipeline" and "Dual pipeline". The "Cache" section shows "Data" as the selected cache type.

**SPECIAL CPU REGISTERS:** The "SPECIAL CPU REGISTERS" section displays various registers and flags:

- PC: 0
- SP: 8096
- SR: 0
- BR: 0
- SR Status Flag: OV, Z, N
- CPU Mode: User (selected), Kernel
- IR: [empty]
- MAR: 2
- MDR: 0

**PROGRAM LIST:** The "PROGRAM LIST" section shows a table of programs:

Name	Base	Start	Type
LOOPTEST	0000	0000	R

**PROGRAM STACK (RAM):** The "PROGRAM STACK (RAM)" section shows a table of program stack entries:

Pos	Val (D)	Addr
-----	---------	------

**Message Dialog:** A message dialog box titled "CPU-OS Simulator" is displayed, stating "Code successfully loaded in memory" with an "OK" button.

**Program Control:** The "Program Control" section includes buttons for "STEP", "RUN", "STOP", "RESET PROGRAM", and "SHOW PCB...". It also features a speed control slider between "Fast" and "Slow".

**Advanced:** The "Advanced" section includes buttons for "COMPILER...", "OS 0...", "INPUT OUTPUT...", "VIRTUAL OS...", and "INTERRUPTS...".

- To enter the OS simulator, click on the **OS 0** button in the current window. The OS window opens.





# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

CPU Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright @ 2006-2013, Besim Mustafa, Edge H...

**PROGRAM SOURCE** pro2.txt

```
PROGRAM Loc
i = 0
for n
i = i
next
end
```

**CPU INSTRUCTIONS IN MEMORY (RAM)**

PAdd	LAdd	Instruction	Bas
0000	0000	MOV #0, R03	000
0006	0006	MOV R03, R01	000
0011	0011	MOV #0, R04	000
0017	0017	MOV R04, R02	000
0022	0022	MOV #40, R04	000
0028	0028	CMP R04, R02	000
0033	0033	JGT 68	000
0037	0037	MOV R01, R03	000
0042	0042	ADD #1, R03	000
0048	0048	MOV R03, R05	000
0053	0053	MOV R05, R01	000
0058	0058	ADD #1, R02	000
0064	0064	JMP 28	000
0068	0068	HLT	000

**Cache - Pipeline** Execution Unit

Pipeline: ☒ Single pipeline ☐ Dual pipeline  
Select pipeline: 0

Cache: Select cache type: Data

**SPECIAL CPU REGISTERS**

PC: 0 SR: 0  
SP: 8096 BR: 0  
SR Status Flag: OV ☐ Z ☐ N ☐  
CPU Mode: User ☒ Kernel ☐  
IR:   
MAR: 2  
MDR: 0

**PROGRAM LIST**

Name	Base	Start	Type
LOOPTEST	0000	0000	R

**PROGRAM STACK (RAM)**

Pos	Val (D)	Addr
-----	---------	------

**COMPILER PROG**

Code genera  
Displaying  
Display com

\*\*\* NOTE: C

Uploading c  
Completed..

**COMPILER**

Edit Source Co

**Program** Instructions Optimize - Assemble

New Program  
Program Name:   
Pages: 1  
Base Address:

Programs  
Program List: LOOPTES  
Base Address:   
☒

**Program Control** CPU View CPU Help

☒ by instruction ☐ by single tick  
 Fast   
 Slow

**Advanced** New CPU

**Register**

Reg V  
  
Show R  
Select R

- You should see an entry, titled **LOOPTEST**, in the **PROGRAM LIST** view. Now that this program is available to the OS simulator. To create as many instances of LOOPTEST process click on **CREATE NEW PROCESS** button as shown below.



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

OS Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright @ 2006-2013, Besim Mustafa, Edge Hill]

### RUNNING PROCESSES

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
-----	------	-------	--------	----------	-------	------	-------	-----	------

WAIT Period: 0 sec. QUEUE KILL Force kill Fault kill Suspend on state change Suspend on pre-emption SHOW MEMORY... SHOW PCB...

### READY PROCESSES (Ready Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
-----	------	-------	--------	----------	-------	------	-------	-----	------

CLEAR REMOVE Suspend on state change SHOW MEMORY... SHOW PCB...

### WAITING PROCESSES (Waiting Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
-----	------	-------	--------	----------	-------	------	-------	-----	------

CLEAR REMOVE RESUME Suspend on state change SHOW MEMORY... SHOW PCB...

### SCHEDULER

Policies Autotun

☒ First-Come, First-Served  
☐ Shortest-Job-First (SJF)  
☐ Round Robin (RR)  
Round Robin and Priority  
RR Time Slice: 5  
0.2

OS Control Views

STEP  
START  
SUSPEND

### PROGRAM LIST

Program Name
LOOPTEST

- Create 4 processes by clicking on the button 4 times. Observe the four instances of the program being queued in the ready queue which is represented by the **READY PROCESSES** view. Make sure the **First-Come-First-Served (FCFS)** option is selected in the **SCHEDULER/Policies** view. At this point the OS is inactive. To activate, first move the



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

**Speed** slider to the fastest position, then click on the **START** button. This should start the OS simulator running the processes.

OS Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright @ 2006-2013, Besim Mustafa, Edge Hill]

### RUNNING PROCESSES

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
-----	------	-------	--------	----------	-------	------	-------	-----	------

WAIT Period: 0 sec. QUEUE KILL Force kill Fault kill ☐ Suspend on state change ☐ Suspend on pre-emption ☐ SHOW MEMORY... SHOW PCB...

### READY PROCESSES (Ready Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
<input type="checkbox"/> 1	LOOPTEST		1	3	0	No	P1		0
<input type="checkbox"/> 2	LOOPTEST		1	3	0	No	P2		0
<input type="checkbox"/> 3	LOOPTEST		1	3	0	No	P3		0
<input type="checkbox"/> 4	LOOPTEST		1	3	0	No	P4		0

CLEAR REMOVE Suspend on state change ☐ SHOW MEMORY... SHOW PCB...

### WAITING PROCESSES (Waiting Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
-----	------	-------	--------	----------	-------	------	-------	-----	------

CLEAR REMOVE RESUME Suspend on state change ☐ SHOW MEMORY... SHOW PCB...

### SCHEDULER

Policies: ☒ First-Come, First-Serve ☐ Shortest-Job-First (SJF) ☐ Round Robin (RR) Round Robin and Priority RR Time Slice: ☒ 5 ☐ 0.2

OS Control: STEP START SUSPEND Views

### PROGRAM LIST

Program Name
LOOPTEST

- Observe the instructions executing in the CPU simulator window. Observe the processes are running which you can see from two views that are **RUNNING PROCESSES** and **READY PROCESSES**





# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

OS Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright © 2006-2013, Besim Mustafa, Edge Hill]

### RUNNING PROCESSES

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
1	LOOPTEST		1	3	0	No	P1	0	0

WAIT Period: 0 sec. QUEUE KILL Force kill ☐ Suspend on state change ☐ Fault kill ☐ Suspend on pre-emption ☐ SHOW MEMORY... SHOW PCB...

### READY PROCESSES (Ready Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
<input type="checkbox"/> 2	LOOPTEST		1	3	0	No	P2		0
<input type="checkbox"/> 3	LOOPTEST		1	3	0	No	P3		0
<input type="checkbox"/> 4	LOOPTEST		1	3	0	No	P4		0

CLEAR REMOVE Suspend on state change ☐ SHOW MEMORY... SHOW PCB...

### WAITING PROCESSES (Waiting Queue)

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid
-----	------	-------	--------	----------	-------	------	-------	-----	------

CLEAR REMOVE RESUME Suspend on state change ☐ SHOW MEMORY... SHOW PCB...

### SCHEDULER

Policies: ☒ First-Come, First-Serve ☐ Shortest-Job-First (SJF) ☐ Round Robin (RR) ☐ Round Robin and Priority (RRP)

Round Robin and Priority (RRP) Time Slice: ☒ 5 ☐ 0.2

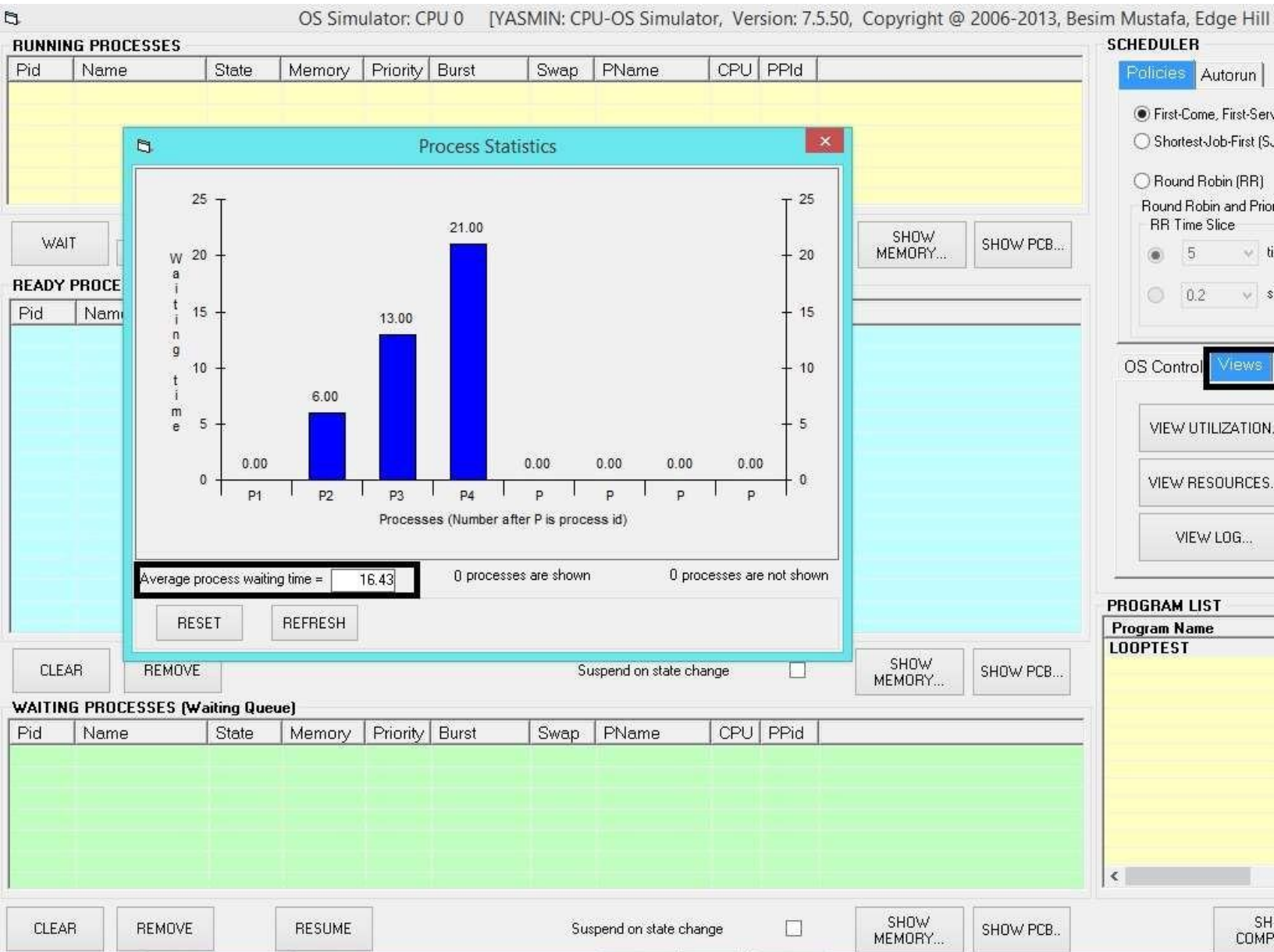
OS Control: STEP STOP SUSPEND Views

### PROGRAM LIST

Program Name: LOOPTEST

- Now, Figure out the average waiting time using the simulator.







# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

### Result:

PROGRAM SOURCE (INPUT)

```
PROGRAM LoopTest
i=0
for n = 0 to 40
i = i + 1
next
for n = 0 to 40
i = i + 1
next
end
```

COMPILER PROGRESS

Pass 1: Tokenizing source statements (Lexical Analysis)

PROGRAM LoopTest: i = 0 for n = 0 to 40 i = i + 1 next

30 tokens are found.  
Tokenizing completed...

Pass 2: Constructing the symbol table  
symbol table constructed...

PROGRAM CODE (OUTPUT)

Label	CPU Instruction	Binary Code	Line	Comments
****	CODE:			
****	MAIN PROG...			
0000	MOV #0, R03	000000000103	0002	Copy the value of 0 to _STe
0006	MOV R03, R01	0001030101	0002	Copy the value of _STempR
0011	MOV #0, R04	000000000104	0003	Copy the value of 0 to _STe
0017	MOV R04, R02	0001040102	0003	Copy the value of _STempR
0022	MOV #40, R04	000000280104	0003	Copy the value of 40 to _ST
0028	CMP R04, R02	3001040102	0003	Compare N with _STempReg
0033	JGT 68	20020044	0003	Jump to code at address 68 i
0037	MOV R01, R03	0001010103	0004	Copy the value of I to _STen
0042	ADD #1, R03	110000010103	0004	Add: _STempReg355 = _STe
0048	MOV R03, R05	0001030105	0004	Copy the value of _STempR
0053	MOV R05, R01	0001050101	0004	Copy the value of _STempR
0058	ADD #1, R02	110000010102	0005	Add: N = N + 1
0064	JMP 28	1D02001C	0005	Jump to code at address 28
0068	MOV #0, R05	000000000105	0006	Copy the value of 0 to _STe
0074	MOV R05, R02	0001050102	0006	Copy the value of _STempR
0079	MOV #40, R05	000000280105	0006	Copy the value of 40 to _ST
0085	CMP R05, R02	3001050102	0006	Compare N with _STempReg
0090	JGT 125	2002007D	0006	Jump to code at address 125
0094	MOV R01, R03	0001010103	0007	Copy the value of I to _STen

PROGRAM SOURCE (INPUT)

```
PROGRAM LoopTest
i=0
for n = 0 to 40
i = i + 1
next
end
```

COMPILER PROGRESS

Code generation completed...

Displaying generated code  
Display completed...

\*\*\* NOTE: Click on numbers in brackets to highlight c

Uploading code to code memory of CPU 0...Please wait  
Completed...

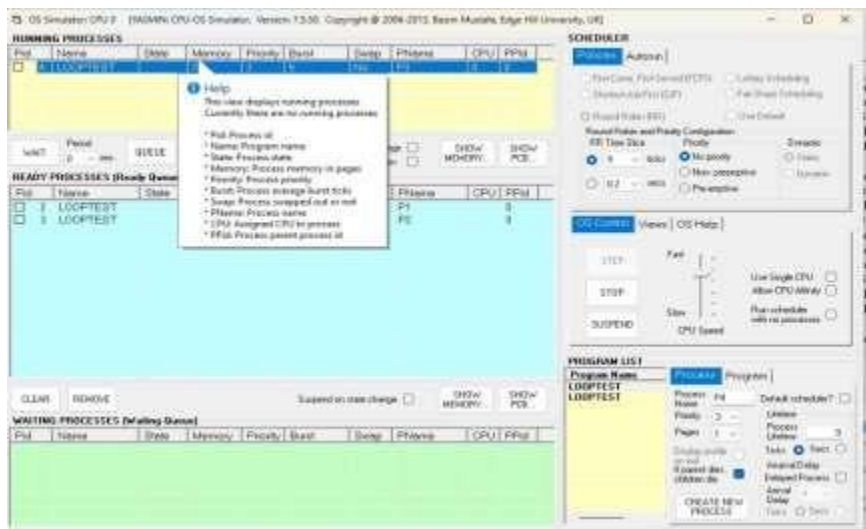
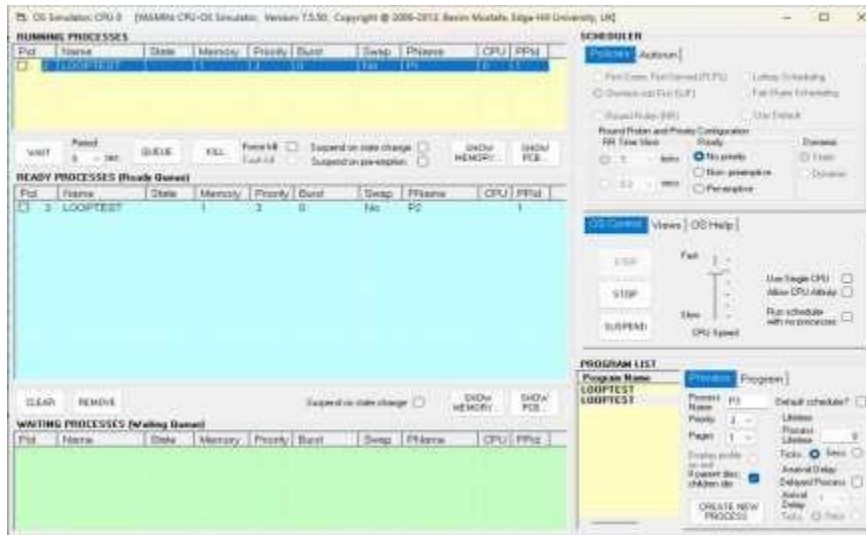
PROGRAM CODE (OUTPUT)

Label	CPU Instruction	Binary Code	Line	Comments
****	CODE:			
****	MAIN PROG...			
0000	MOV #0, R03	000000000103	0002	Copy the value of 0 to _STemp
0006	MOV R03, R01	0001030101	0002	Copy the value of _STempReg
0011	MOV #0, R04	000000000104	0003	Copy the value of 0 to _STemp
0017	MOV R04, R02	0001040102	0003	Copy the value of _STempReg
0022	MOV #40, R04	000000280104	0003	Copy the value of 40 to _STen
0028	CMP R04, R02	3001040102	0003	Compare N with _STempReg2
0033	JGT 68	20020044	0003	Jump to code at address 68 if
0037	MOV R01, R03	0001010103	0004	Copy the value of I to _STempl
0042	ADD #1, R03	110000010103	0004	Add: _STempReg235 = _STen
0048	MOV R03, R05	0001030105	0004	Copy the value of _STempReg
0053	MOV R05, R01	0001050101	0004	Copy the value of _STempReg
0058	ADD #1, R02	110000010102	0005	Add: N = N + 1
0064	JMP 28	1D02001C	0005	Jump to code at address 28 (U
0068	HLT	2F	0006	Stop simulation
****	DATA:			



# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

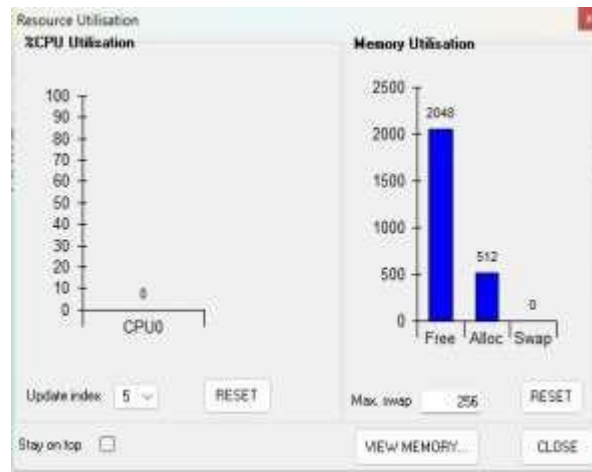




# Vidyavardhini's College of Engineering & Technology

## Department of Computer Engineering

PID	Name	PPID	Pri	CPUT	STA	MEM	CPU	ETM
2	LOOPTEST	0	3	65	Running	1	0	4889
3	LOOPTEST	1	3	0	Ready	1		4889



**System Resources**

Allocating resources for process id:

**Resource List**

Resource	Used by	Requested by	Allocate	Block	Release
R0			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
R1			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
R2			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
R3			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
R4			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
R5			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Resource colour key: ■ Available ■ Allocated only ■ Allocated + Requested

**Prevent** ☐ Disable hold and wait ☐ Disable circular wait ☐ Use total ordering

**Recover** ☐ Abort processes ☐ Pre-empt resources

**Avoid** ☐ Enable

**Detect** ☐ Do not detect ☐ Every 1 sec ☐ Randomly ☐ CPU Utilization < 50 % ☒ After every alloc. and de-alloc.

Buttons: SHOW DEADLOCKED PROCESSES... DEADLOCK Count: 0 RESET RELEASE ALL CLOSE



**Conclusion:** In conclusion, the CPU-OS simulator for process scheduling algorithms provides a valuable platform for studying, testing, and optimizing various scheduling strategies. It enables researchers and practitioners to analyze the performance of algorithms in a controlled environment, facilitating informed decision-making and driving innovation in operating system design.