Bansilal Ramnath Agarwal Charitable Trust‘s

Vishwakarma Institute of Technology

Bibwewadi, Pune

Accredited with ‘A++’ Grade by NAAC

Presentation

On

Subject: Operating Systems (CS2008)

Course Project (Phase 2)

by

Prof. Archana Burujwale

Department of Computer Engineering

CS2008:: Operating Systems Laboratory

Lab: 2 Hours/Week

CS2008:: Operating Systems

Course Prerequisites: Computer Architecture & organization, Data Structure

Course Objectives:

1.To learn functions of Operating System

2.To learn the importance of concurrency and how to implement concurrent abstractions correctly in an OS.

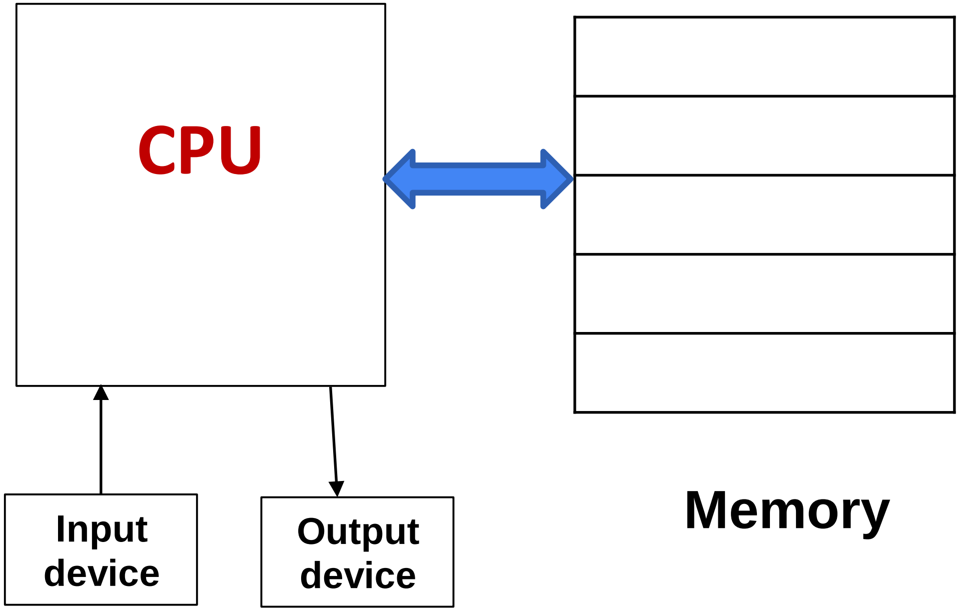
3.To learn OS scheduling policies and mechanisms.

4.To deal with deadlock

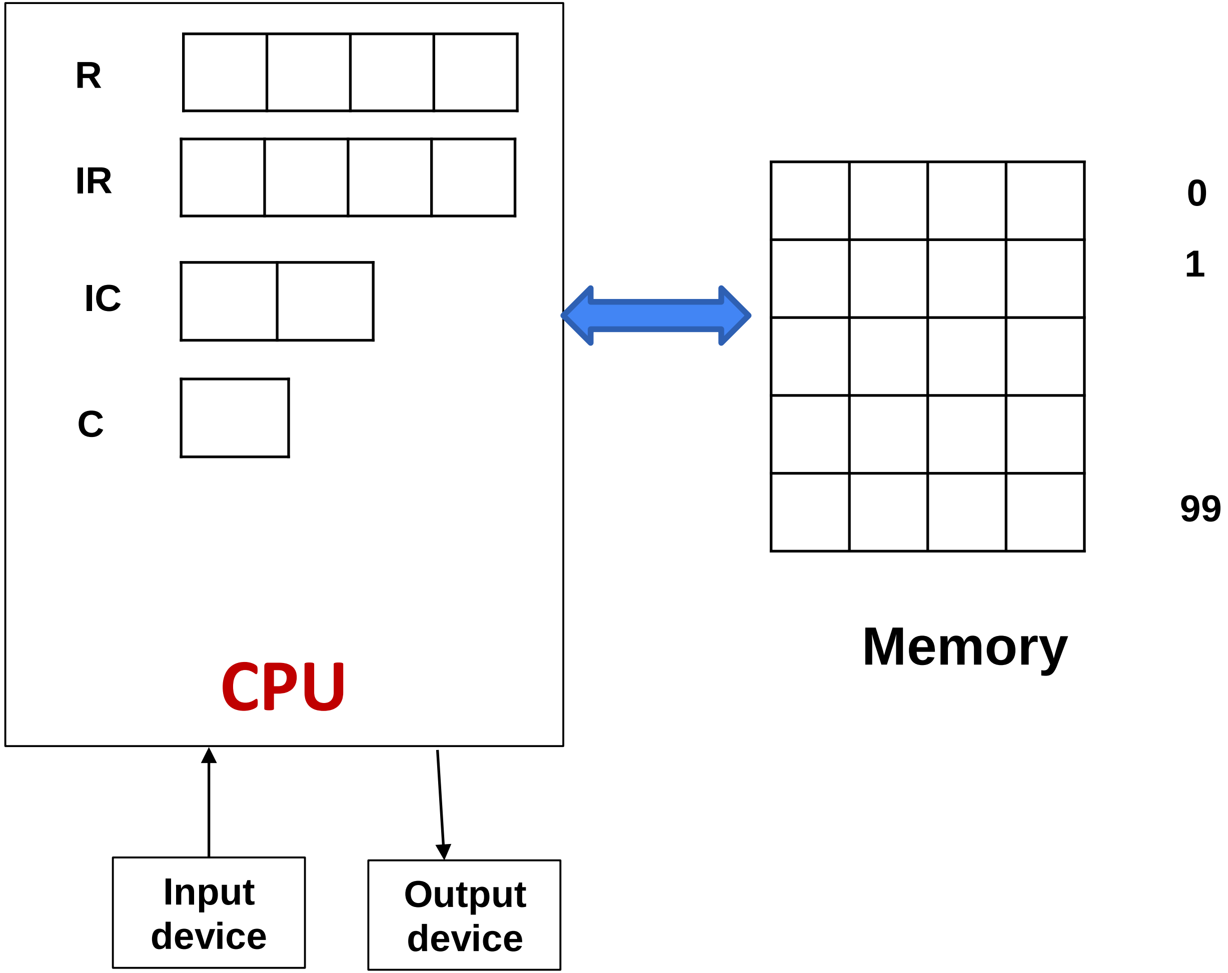
5.To learn memory management schemes in various ways to improve performance, and how this impacts system complexity

6. To learn design & develop the Operating system from a scratch.

Scenario:



Scenario:



Specifications

- Main memory: 100 words * 4 byes
- Block: 10 words * 4 bytes
- Input device: CPU can read multiple cards
- Card reader: It can read multiple cards
- Size of each card: 1 block: 10* 4 bytes= 40 bytes
- Output device: eg. Line printer
- Can print one line = 1 card = 40 bytes
- Size of each card: Max. 40 bytes
- Except H all other instructions are of 4 bytes.

Types of cards

- Control card
- Program Card
- Data Card

Instruction Set

- GD <10>: Get data from the data card and put it in the memory block whose starting address is 10
- PD <10>: Print the data from block whose starting address is 10 and print it in output file. It prints complete block.
- LR <10>: Load register with the contents of memory location <10>. Register is general purpose register.
- SR <10>: Store the contents of register to memory location <10>
- CR <10>: Compare content of register R and memory location <10>. Result will be stored in Toggle register, if both values are equal: Toggle =True (T)
- BT <05>: Branch on Toggle

If toggle is true, jump to memory location <05> and start executing instructions from this location.

- H: Halt: stop the execution

Unit-I Multiprogramming Operating System (MOS) Project
(Second Phase)

Assumptions:

- Jobs may have program errors
- PI interrupt for program errors introduced
- No physical separation between jobs
- Job outputs separated in output file by 2 blank lines
- Paging introduced, page table stored in real memory
- Program pages allocated one of 30 memory block using random number generator
- Load and run one program at a time
- Time limit, line limit, out-of-data errors introduced
- TI interrupt for time-out error introduced
- 2-line messages printed at termination

Memory

Main Memory

Block	Memory Location	Entry
0	0..9	
1	10-19	
2	20-29	
3	30-39	
4		
…..		
29	290-299	

Virtual Memory

Block	Memory Location	Entry
1	0..9	
2	10-19	
3	20-29	
4	30-39	
…..		
9	90-99	

Types of Errors

1. Opcode error
2. Operand error
3. Time limit exceeded
4. Line limit exceeded
5. Out of data
6. Page fault: valid, invalid

Sample Program

```
$AMJ 0001 0003 0001  ;Job ID: 0001, Total Time limit: 0003, Total line limit: 0001
GD 10 PD 10 H
$DTA
Hello World
$END 0001
```

1. Opcode Error

Here we consider 7 instructions. If opcode is other than this, this is opcode error.

Eg:

Valid Instruction: GD 10

Opcode Error: XD 10

Handling: Error message displayed on screen and in output file; program terminates

2. Operand Error

- Valid range of addresses is 0 to 99. Hence, If the operand is not in this range, operand error

3. Time limit exceeded

- **$AMJ 0001 0003 0001**
- GD and SR instructions will take 2 units of time: because for both instructions we get page fault. Hence,

  1 unit: to handle page fault

  1 unit: to execute the instruction
- All other instructions will take 1 unit of time.

$AMJ 0001 0004 0001 ;Job ID: 0001, Total Time limit: 0004, Total line limit: 0001

GD 10 PD 10 H

$DTA

Hello World

$END 0001

3. Time limit exceeded

- Set one variable which will work as timer. The timer will be incremented after execution of every instruction.
- **Error detection:**

If the timer count (TC) is matching with “Total time limit (TTL)”; no error

If TC> TTL; Time limit exceed error

4. Line limit exceeded

$AMJ 0001 0004 0001

Total Line Limit: How many lines program going to print

If number of PD instructions > TLL; Line limit exceeded error

Handling:

Set one line limit counter and check its value against TLL

5. Out of data error

This error is related to number of GD instructions and the number of data cards present in the job.

If no. of GD instructions> actual data; Out of data error

Paging

Virtual Memory

Block	Memory Location	Entry
0	0..9	
1	10-19	
2	20-29	
3	30-39	
…..		
9	90-99	

Page Table

(present in Main memory block)

Block From Virtual memory	Block from main memory
0	13
1	16
	…

Main Memory

Block	Memory Location	Entry
0	0..9	
1	10-19	
2	20-29	
3	30-39	
4		
…..		
29	290-299	

6. Page Fault:

- **Valid Page Fault:**
- In case of GD and SR instructions’ execution, the data card will not be present in memory, so page fault may occur.
- In this case, the data card need to be loaded in the memory, IC = IC-1; Now execute GD instruction.

- **Invalid Page fault**:
- In other instructions, it is invalid page fault; Error need to be displayed.
- For PD, we have invalid page fault.

Load Function

- Instruction:
- **$AMJ 0001 0004 0001**
- **Action:**

1. Create PCB

2. Create Page table in main memory

PCB Creation

- PCB will be created when we recognize $AMJ i.e. Start of the job.
- Create one structure variable which contains:
- Job ID: From line $AMJ
- Total Time Limit (TTL): From line $AMJ
- Total Line Limit (TLL): From line $AMJ
- Total time count (TTC): Initialize to 0
- Line Limit count (LLC): Initialize to 0

Page table creation

1. **Creating Page table:** The page table will reside in the main memory. So, we need to assign one block of main memory as page table.

Procedure:

1. Select any random number between 0 to 29;

Eg: random no: 20

Block no. 20 will be assigned as page table in main memory.

The real address of the block will be 20*10 = 200 to 209

2. PTR =20

Main Memory

Block	Memory Location	Entry
20	200	****
	201	****
	202…	****
	209	****
29	290-299	

Program Card 1

- Generate random number and consider it as block number in memory.
- If the block is not occupied, make its entry in page table and load the list of instructions in the block.
- Using random number generator, Select the page for the job, if the page is not occupied.
- Eg: Random number = 14

Main Memory

Block	Memory Location	Entry
14	140	GD 10
	141	PD 10
	142..	H
20	200	**14
	201	****
	202…	****

Program Card 2

- Generate random number and consider it as block number in memory.
- If the block is not occupied, make its entry in page table and load the list of instructions in the block.
- Using random number generator, Select the page for the job, if the page is not occupied.
- Eg: Random number = 23

Main Memory

Block	Memory Location	Entry
14	140	GD 10
	141	PD 10
	142..	H
20	200	**14
	201	**23
	202…	****
23	230	…
	231	…

Program Card 3

- Generate random number and consider it as block number in memory.
- If the block is not occupied, make its entry in page table and load the list of instructions in the block.
- Using random number generator, Select the page for the job, if the page is not occupied.
- Eg: Random number = 9

Main Memory

Block	Memory Location	Entry
9	90	…
	…99	…
14	140	GD 10
	141	PD 10
	142..	H
20	200	**14
	201	**23
	202…	**09
23	230	…
	231	…

Execution of the Program

- Virtual address to main memory address calculation:
- Initially, IC=0
- Hence, Virtual address (VA) = 0

```
Function Add_map(VA)
  PTE = PTR + VA/10
  RA = M[PTE] *10  + VA %10
```

Example:

```
 VA= 0
 PTE = 200 + 0/10 = 200+0 = 200
RA = M[200] *10 + 0 %10
       = 14 *10 + 0 = 140;
```

At location 140, first instruction of job need to be loaded

Interrupts

- Program Interrupt (PI)

PI = 1: Opcode error
= 2: Operand error
=3: Page fault

- Timer Interrupt (TI)

TI = 0 : No error
= 1: Time limit exceeded error

In case of any error, two lines need to be printed at output:

Line 1: Values of IC, IR, TTC, TTL, LLC,

Line 2: Program terminated abnormally because of _____ error

Notation

- M: memory; IR: Instruction Register (4 bytes)
- IR [1, 2]: Bytes 1, 2 of IR/Operation Code
- IR [3, 4]: Bytes 3, 4 of IR/Operand Address
- M[&]: Content of memory location &
- IC: Instruction Counter Register (2 bytes)
- R: General Purpose Register (4 bytes)
- C: Toggle (1 byte)

Notation

- PTR: Page Table Register (4 bytes)
- PCB: Process Control Block (data structure)
- VA: Virtual Address
- RA: Real Address
- TTC: Total Time Counter
- LLC: Line Limit Counter
- TTL: Total Time Limit
- TLL: Total Line Limit
- EM: Error Message
- ← : Loaded/stored/placed into

Interrupt values

SI = 1 on GD

= 2 on PD

= 3 on H

TI = 2 on Time Limit Exceeded

PI = 1 Operation Error

= 2 Operand Error

= 3 Page Fault

Error Message Coding

EM	**Error**
0	No Error
1	Out of Data
2	Line Limit Exceeded
3	Time Limit Exceeded
4	Operation Code Error
5	Operand Error
6	Invalid Page Fault

Start of Program

BEGIN

INITIALIZATION

SI = 3, TI = 0

Load Function

```
LOAD
While not e-o-f
        Read next (program or control) card from input file in a buffer
        Control card: $AMJ, create and initialize PCB
                      ALLOCATE (Get Frame for Page Table)
                      Initialize Page Table and PTR
                      Endwhile
                      $DTA, STARTEXECUTION
                      $END, end-while
```

Load Function

```
            Program Card: ALLOCATE (Get Frame for Program Page)
                          Update Page Table
                          Load Program Page in Allocated Frame
                         End-While
End-While
STOP
```

Start execution Function

```
STARTEXECUTION
IC ← 00
EXECUTEUSERPROGRAM
END (MOS)
```

```
EXECUTEUSERPROGRAM (SLAVE MODE)
ADDRESS MAP (VA, RA)
        Accepts VA, either computes & returns RA or sets PI $\leftarrow$ 2 (Operand Error) or PI $\leftarrow$ 3 (Page Fault)
LOOP
        ADDRESSMAP (IC, RA)
        If PI $\neq$ 0, End-LOOP (F)
        IR $\leftarrow$ M[RA]
        IC $\leftarrow$ IC+1
        ADDRESSMAP (IR[3,4], RA)
        If PI $\neq$ 0, End-LOOP (E)
        Examine IR[1,2]
            LR:    R $\leftarrow$ M [RA]
            SR:    R $\rightarrow$ M [RA]
            CR:    Compare R and M [RA]
                   If equal C $\leftarrow$ T else C $\leftarrow$ F
            BT:    If C = T then IC $\leftarrow$ IR [3,4]
            GD:    SI = 1 (Input Request)
            PD:    SI = 2 (Output Request)
            H:     SI = 3 (Terminate Request)
            Otherwise PI $\leftarrow$ 1 (Operation Error)
        End-Examine

End-LOOP (X)       X = F (Fetch) or E (Execute)
```

```
SIMULATION
        Increment TTC
        If  $TTC = TTL$ then $TI \leftarrow 2$

If SI or PI or $TI \neq 0$ then Master Mode, Else Slave Mode
```

MOS (MASTER MODE)

Case TI and SI of

TI	SI	Action
0	1	READ
0	2	WRITE
0	3	TERMINATE (0)
2	1	TERMINATE (3)
2	2	WRITE, THEN TERMINATE (3)
2	3	TERMINATE (0)

Case TI and PI of

TI	PI	Action
0	1	TERMINATE (4)
0	2	TERMINATE (5)
0	3	If Page Fault Valid, ALLOCATE, update page Table, Adjust IC if necessary, EXECUTE USER PROGRAM OTHERWISE TERMINATE (6)
2	1	TERMINATE (3,4)
2	2	TERMINATE (3,5)
2	3	TERMINATE (3)

```
READ
    If next data card is $END, TERMINATE (1)
    Read next (data) card from input file in memory locations RA through RA + 9
    EXECUTEUSERPROGRAM

WRITE
    $LLC \leftarrow LLC + 1$
    If $LLC > TLL$, TERMINATE (2)
    Write one block of memory from locations RA through RA + 9 to output file
    EXECUTEUSERPROGRAM

TERMINATE (EM)
    Write 2 blank lines in output file
    Write 2 lines of appropriate Terminating Message as indicated by EM
    LOAD
```

Thank You