

# Component 1

---

- Basics of Fortran programming
- Introduction to HPC architecture
- Parallel computing – hardware and software
- Message Passing Interface (MPI)
- Accelerators (GPU) (if time permits)

# Component 2

---

- Simulation techniques
  - Force fields
  - Molecular dynamics
  - Monte Carlo method
  - Free energy techniques (if time permits)

# Component 3

---

- Optimizing the cluster operation: Jobs scheduler, load cluster level load balancing, etc
- Installation of software on HPC machines
- Parallelization, communication and load balancing
- Domain decomposition, multiprocessor communication, dynamic load balancing
- MD hands-on
- MC hands-on
- Visualization and analysis

# Recommended textbooks

---

- An Introduction to Parallel Programming by Peter S. Pacheco
- Using MPI: portable parallel programming with the message-passing interface:  
William Gropp, Erwing Lusk, Anthony Skjellum
- Computer Architecture: A Quantitative Approach: John L. Hennessy and David A.  
Patterson
- Understanding Molecular Simulations: Frenkel and Smit
- Computer Simulations of Liquids: M. P. Allen and D. J. Tildesley
- Molecular Modeling and Simulation: Tamar Schlick

# Evaluation of CD61004 subject

---

Weight 40%

Periodic hands-on tests/home assignments (15%); project (10%); attendance/others (5%); class tests (10%)

Weight 60%

Mid-semester exam: 30% ; End-semester exam: 30%

# Topic: Fortran Programming

---

## Objectives

- How to compile Fortran code
- Basics of Fortran programming

# Topic: Fortran Programming

---

## Low level language vs high level language

### Program written in assembly language to print 'hello world'

```
section .data                ;.data starts here
    msg db 10d,13d,"Hello World " ;String gets initialized
    l equ $-msg              ;Length Of String
section .text                ;.text starts here
    global _start            ;Moving to _start
_start:                      ;_start label
    mov rax,1                ;Sys_Write Function
    mov rdi,1                ;Std_Out File Descriptor
    mov rsi,msg              ;Offset of msg
    mov rdx,l                ;Length Of msg
    syscall                  ;Call the Kernel
    mov rax,60               ;Sys_Exit Function
    mov rdi,0                ;Sucessful Termination
    syscall                  ;Call The Kernel
end:                          ;end Label
```

# Topic: Fortran Programming

---

## Why Fortran

- Easy
- Fast
- Oldest programming language still being used
- General purpose, high-level programming language developed in 1957 for numeric and scientific computing (engineering applications)
- Fortran stands for *Formula Translation*
- Many supercomputing applications are written in Fortran and still being in usage



# Topic: Fortran Programming

---

## Fortran programming

- All program names should end with “.f90”.
- How to compile a Fortran program
  - *compilers: gfortran, ifort, etc*
  - *gfortran program.f90 -o program.x*
  - *How to install gfortran in ubuntu:*
    - `sudo apt update`
    - `sudo apt install gfortran`
- NOT case-sensitive
- Line starting with “!” are treated as comment line.

# Topic: Fortran Programming

## Text editor

```
(base) sandeep@raakshasi $ gedit p1.f90 &  
[1] 18655  
(base) sandeep@raakshasi $
```



# Topic: Fortran Programming

```
program test
```

No output

```
end program test
```

# Topic: Fortran Programming

```
program test
```

```
  write(*,*) ' Hello World'
```

```
end program test
```

Output

Hello World

Printing data to stdout

# Topic: Fortran Programming

```
program test
```

```
  write(*,*) ' Sum ', 27 + 23 + 22
```

```
end program test
```

Output

Sum 72

Sum of three numbers

# Topic: Fortran Programming

```
program test
```

```
  is1 = 27
```

```
  is2 = 23
```

```
  is3 = 22
```

```
  write(*,*) ' Sum ', is1 + is2 + is3
```

```
end program test
```

Output

Sum 72

Using variables to do the sum

# Topic: Fortran Programming

```
program test
```

Output

Sum    72

```
is1 = 27
```

```
is2 = 23
```

```
is3 = 22
```

```
write(*,*) ' Sum ', is1 + is2 + is3
```

```
end program test
```

- Rules for variable names:
  - It must start with alphabet. Rest of the name can have both letters (a-z), number and underscore(\_) character
  - Space or blank character is not allowed

# Topic: Fortran Programming

```
program test
```

```
  is1 = 27
```

```
  is2 = 23
```

```
  is3 = 22
```

```
  itotal = is1 + is2 + is3
```

```
  write(*,*) ' Sum ', itotal
```

```
end program test
```

Output

Sum 72

Improve the readability



# Topic: Fortran Programming

```
program test
```

```
  s1 = 27
```

```
  s2 = 23
```

```
  s3 = 22
```

```
  total = s1 + s2 + s3
```

```
  write(*,*) ' Sum ', total
```

```
end program test
```

Output

Sum 72.00000000

Notice the difference in output

# Topic: Fortran Programming

```
program test
  implicit none

  integer :: s1, s2, s3, total

  s1 = 27
  s2 = 23
  s3 = 22

  total = s1 + s2 + s3

  write(*,*) ' Sum ', total
end program test
```

Output  
Sum 72

Declaring the variables

# Topic: Fortran Programming

```
program test
  implicit none

  integer :: s1, s2, s3, total

  s1 = 27

  s2 = 23

  s3 = 22.5

  total = s1 + s2 + s3

  write(*,*) ' Sum ', total

end program test
```

Output

Sum	72
-----	----

Sum is not correct

# Topic: Fortran Programming

FORTRAN program has FOUR elements

```
program test
implicit none

integer :: s1, s2, s3, total
s1 = 27
s2 = 23
s3 = 22.5
total = s1 + s2 + s3
write(*,*) ' Sum ', total

end program test
```

Program name

Declaration and initialization of variables

Main body of the program

Subprogram(s)

Structure of the FORTRAN program

# Topic: Fortran Programming

FORTRAN program has FOUR elements

```
program test
implicit none

integer :: s1, s2, s3, total
s1 = 27
s2 = 23
s3 = 22.5
total = s1 + s2 + s3
write(*,*) ' Sum ', total
end program test
```

Program name

Declaration and initialization of variables

The available data types are,

- real (kind=8)::
- integer ::
- complex ::
- character(len=100) ::
- logical ::

subprogram(s)

Structure of the FORTRAN program

# Topic: Fortran Programming

## IF conditional statement

```
if (logical expression 1) then
    ! block 1
else if (logical expression 2) then
    ! block 2
-----
-----
-----
else
    ! block 3
end if
```

# Topic: Fortran Programming

## IF conditional statement

```
if (logical expression 1) then
    ! block 1
else if (logical expression 2) then
    ! block 2
-----
-----
-----
else
    ! block 3
end if
```

Operator	Alternative	Meaning
.eq.	==	equal to
.ne.	/=	not equal to
.lt.	<	less than
.le.	<=	less than or equal to
.gt.	>	greater than
.ge.	>=	greater than or equal to
.and. .or. .not.		boolean expressions

# Topic: Fortran Programming

---

## IF conditional statement

```
if (s1 > s2) then
    write(*,*) s1," is greater than ",s2
else if (s1 > s3) then
    write(*,*) s1," is greater than ",s3
else
    write(*,*) s1," is smallest among all numbers"
end if
```



# Topic: Fortran Programming

---

Write a program using *if statements* to find a largest of three numbers

Write a program to calculate the area of a circle

Fortran emulator: [https://www.tutorialspoint.com/compile\\_fortran\\_online.php](https://www.tutorialspoint.com/compile_fortran_online.php)

# Topic: Fortran Programming

---

## FORTRAN – Reading material

- Please go through this FORTRAN program for a quick overview,  
<https://learnxinyminutes.com/docs/fortran95/>
- Please go through this document for quick overview of FORTRAN  
<https://www.ideo.columbia.edu/~mspieg/mmm/Fortran.pdf>
- Book: Computer Programming in Fortran 90 and 95, V. Rajaraman
- Tutorial on Fortran along the emulator,  
[https://www.tutorialspoint.com/fortran/fortran\\_overview.htm](https://www.tutorialspoint.com/fortran/fortran_overview.htm)