

---

# High Performance Computing

## Lecture 4

Matthew Jacob

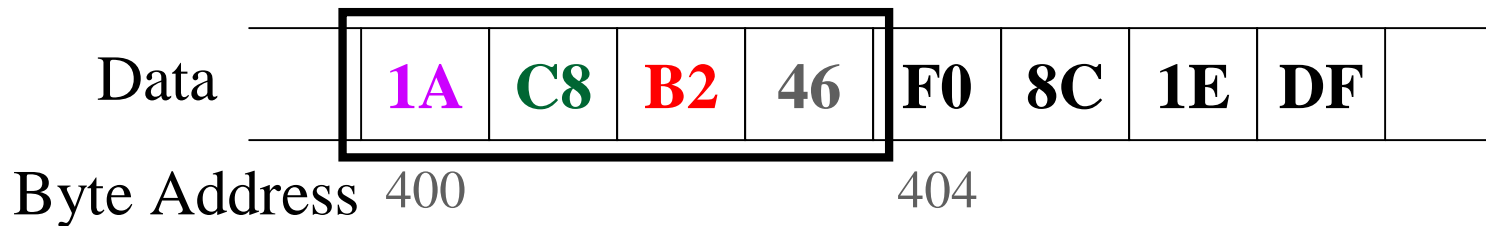
Indian Institute of Science

---

# Terms: Word Size, Word Alignment

The **Word Size** of a computer

The size of an integer or pointer on the computer  
32b (4B) on many machines



## Word Alignment

‘Integer variable X is not word aligned’

The data item is not located at a word boundary

Word boundaries: addresses 0, 4, 8, 12, ...

---

# Instruction Set Architecture (ISA)

View of the computer that is visible to the programmer (or compiler)

There are 2 styles of instruction set design

## 1. CISC

- Complex Instruction Set Computer
- A single instruction performs a complex operation involving several actions

## 2. RISC

- Reduced Instruction Set Computer
- Each instruction performs only a simple operation

---

# Instruction Set Architecture

- Description of machine from view of the programmer/compiler
  - Example: Intel<sup>®</sup> x86 Instruction Set Architecture
- Includes specification of
  1. The different kinds of instructions available ([instruction set](#))
  2. How operands are specified ([addressing modes](#))
  3. What each instruction looks like ([instruction format](#))

---

# Kinds of Instructions

## 1. Arithmetic/logical instructions

- ❑ add, subtract, multiply, divide, compare (int/fp)
- ❑ or, and, not, xor
- ❑ shift (left/right, arithmetic/logical), rotate

---

# Kinds of Instructions

1. Arithmetic/logical instructions
2. Data transfer instructions
  - ❑ load (copy data to a register from memory)
  - ❑ store (copy data to a memory location from a register)
  - ❑ move

---

# Kinds of Instructions

1. Arithmetic/logical instructions
2. Data transfer instructions
3. Control transfer instructions
  - jump, conditional branch, function call, return

---

# Kinds of Instructions

1. Arithmetic/logical instructions
2. Data transfer instructions
3. Control transfer instructions
4. Other instructions
  - e.g., halt



---

# Operand Addressing Modes

- Operands to an instruction
  - **Source**: input value to instruction
  - **Destination**: where result is to go
- Addressing Mode
  - How the location of an operand to an instruction is specified
- An operand can be either
  - in a memory location
  - in a register

---

# Addressing Modes: Operand in Register

## 1. Register Direct Addressing Mode

Operand is in the specified general purpose register

### Example

Suppose that the General Purpose Registers numbered as R0, R1, R2, etc

R1	67
R2	24
R3	37

ADD R1, R2, R3 / R1 = R2 + R3

destination operand      source operands

Information that must be present in the instruction:

Operation (e.g., add)

Identities of source and destination registers

---

# Addressing Modes: Operand in Register

1. **Register Direct** Addressing Mode
2. **Immediate** Addressing Mode

Operand is included in the instruction

ADD R1, R2, 7                      /R1 = R2 + 7

Information that must be present in the instruction:

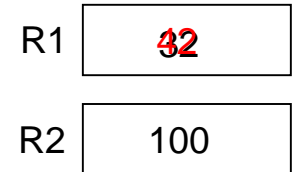
The “immediate operand” value

# Addressing Modes: Operand in Memory

## 3. Register Indirect Addressing Mode

Memory address of operand is in the specified general purpose register

ADD R1, R1, (R2)



Address	96	100	104	108
Value	0	10	35	-17

MAIN MEMORY

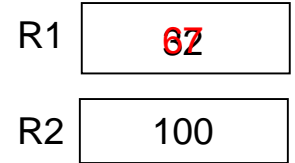
Information that must be present in the instruction:

Identity of register which contains the operand address

# Addressing Modes: Operand in Memory

- 3. Register Indirect Addressing Mode
- 4. Base-Displacement Addressing Mode

Memory address of operand is calculated as the sum of value in specified register and specified displacement



```
ADD R1, R1, 4(R2)
```

Address	96	100	104	108
Value	0	10	35	-17

MAIN MEMORY

Information that must be present in instruction:

Identity of base register (e.g., R2)

Value of displacement to be added (e.g. 4)

---

# Addressing Modes: Operand in Memory

3. **Register Indirect** Addressing Mode
4. **Base-Displacement** Addressing Mode
5. **Absolute** Addressing Mode

Memory address of operand is specified directly in the instruction

```
ADD R1, R2, #100
```