# 

COMPUTER ORGANIZATION

NAME: SANDHIYA L

ROLL NO: 660416

REG NO:917723IT130

SUBJECT CODE:22IT230

COURSE:B.TECH IT

ADDRESSING MODES

The different ways of specifying the locations of an operand in an instruction are called as addressing modes.Addressing modes dictate how instruction access data in a computer memory.Programs operate on data residing in the computers memory .Data can be organized in various ways such as lists and arrays.Addressing modes specify how instruction operands locations are specified.

Types of Addressing Modes:

In computer oragnization, there are following types of addressing modes,

1. Immediate Addressing Mode
2. Register Addressing Mode
3. Absolute Addressing Mode
4. Register Indirect Addressing Mode
5. Indexed Addressing Mode
6. Base Register Addressing Mode

1.IMMEDIATE ADDRESSING MODE:

In this addressing mode, the operand is specified in the instruction explicitly. Instead of an address field, an operand field is present that contains the operand.

**Operand Value**: In immediate addressing mode, the operand is a constant value or a literal value that is part of the instruction

**Usage**: Immediate addressing mode is often used when the operand is a constant value that does not change frequently during program execution.

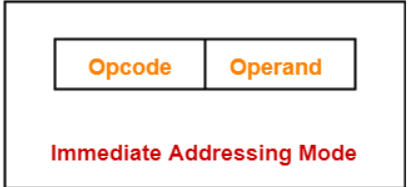
**Advantages**: It reduces memory access time since the operand value is readily available within the instruction, which can lead to faster execution.

**Limitations**: Immediate addressing mode is limited by the size of the instruction format, as larger constants may not fit within the instruction. Additionally, it's not suitable for operations where the operand value needs to be modified during program execution.

Overall, immediate addressing mode is efficient for operations involving constant values and helps in speeding up the execution of instructions.

Examples- • Load R1, #1000 is interpreted as R1 ← 1000

• ADD R2, #3 is interpreted as R2 ← [R2] + 3



Example 2:

• Add R4,R6,#200

• Adds the value of 200 to the contents of register,R6 and places the result into register to R4.

# REGISTER ADDRESSING MODE

In this addressing mode,

• The operand is contained in a register set.

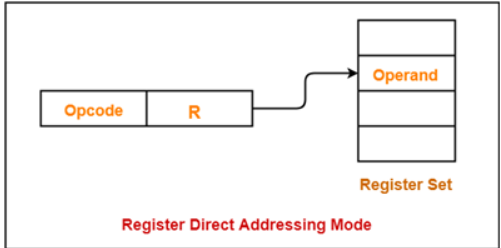
•The operand is the contents of a processor register.

• The address field of the instruction refers to a CPU register that contains the operand.

• No reference to memory is required to fetch the operand.

• This addressing mode is similar to direct addressing mode.

• The only difference is that the address field of the instruction refers to a CPU register instead of main memory



Example-

• ADD R will increment the value stored in the accumulator by the content of register R.

AC ← AC + [R]

• ADD R1, R2 is interpreted as R1 ← [R1] + [R2]

• Load R1, R2 is interpreted as R1 ← [R2]

Example 2:

•ADD R4,R2,R3

1. **Usage**: Register direct addressing mode is commonly used for operations where the operands are stored in registers and need to be manipulated directly.
2. **Advantages**: This mode can be very efficient because accessing data from registers is typically faster than accessing data from memory. It also allows for compact instruction formats since only register numbers need to be specified.
3. **Limitations**: Register direct addressing mode is limited by the number of available registers in the CPU architecture. Additionally, it may not be suitable for operations involving operands that are not stored in registers.

Overall, register direct addressing mode is efficient for operations involving register data and contributes to faster execution of instructions.

Top of Form

#ABSOLUTE ADDRESSING MODE

•The operand is in a memory location.

•the address of this location is given explicitly in the instruction

1. **Usage**: Absolute addressing mode is utilized when the exact memory location of the operand is known and fixed at the time of coding or assembly.
2. **Advantages**: It provides direct access to memory locations, which can be beneficial for accessing specific data stored at known memory addresses.
3. **Limitations**: Absolute addressing mode can be inflexible because if the memory location of the operand changes, the instruction needs to be modified accordingly. It's also not suitable for programs that need to be loaded into different memory locations

Overall, absolute addressing mode offers direct access to memory locations but lacks flexibility compared to other addressing modes that allow for more dynamic memory referencing.

Top of Form

Example 1:

•Load R2,LOC

•loads the value in the memory location LOC into register R2.

Example 2:

•Load R2,NUM1

# REGISTER INDIRECT ADDRESSING MODE:

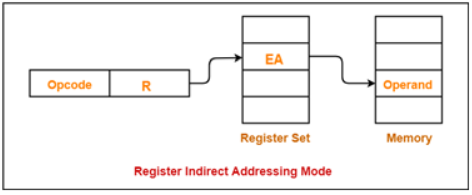
In this addressing mode,

• The address field of the instruction refers to a CPU register that contains the effective address of the operand.

• Only one reference to memory is required to fetch the operand.

• This addressing mode is similar to indirect addressing mode.

• The only difference is the address field of the instruction refers to a CPU register.



1. **Usage**: Register indirect addressing mode is commonly used when the exact memory location of the operand may vary during program execution or when accessing data structures like arrays or linked list
2. **Advantages**: It provides flexibility because the memory address can be updated dynamically in the register, allowing for more versatile memory access.
3. **Limitations**: Register indirect addressing mode may require extra instructions to load the memory address into the register before accessing the operand, potentially increasing execution time.

Overall, register indirect addressing mode offers flexibility in accessing memory locations and is suitable for scenarios where the memory address of the operand needs to be determined dynamically at runtime.

Top of Form

Example-

• ADD R will increment the value stored in the accumulator by the content of memory location specified in register R.

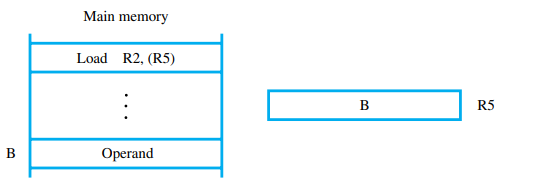
AC ← AC + [[R]]

• Load R1, @R2 is interpreted as R1 ← [[R1]]

• ADD R1, @(R2) is interpreted as R1 ← [R1] + [[R2]]

• ADD R1, (R2) is interpreted as R1 ← [R1] + [[R2]]

Example 2: •Load R2,(R5)



# INDEXED ADDRESSING MODE

In this addressing mode,

• Effective address of the operand is obtained by adding the content of index register with the address part of the instruction.

Effective Address = Content of Index Register + Address part of the instruction

Index mode symbolically as

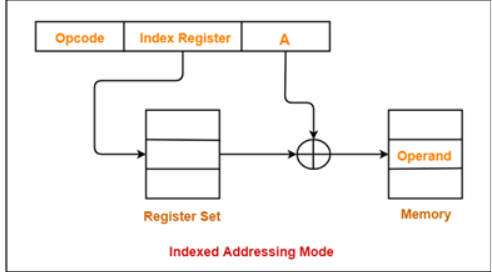
X(Ri)

where X denotes a constant signed integer value contained in the instruction and Ri is the name of the register involved.

The effective address of the operand is given by

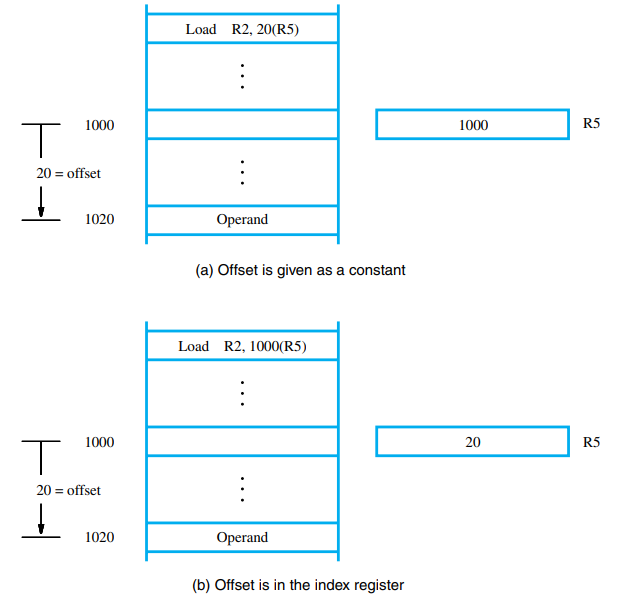
EA = X + [Ri]

The contents of the register are not changed in the process of generating effective address.



Example:

Load R2,20(R5)



Advantage:

Flexibility:

allows for efficient acces to array elements by using the index register as a base address,enabling easy iteration through arrays.

Limitation:

Limited Range:

The constant displacement may limit the range of address that can be accesed efficiently.

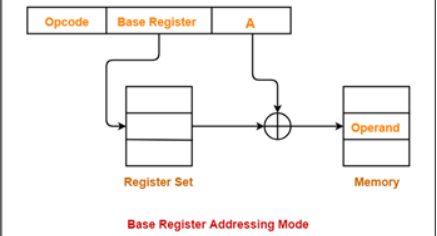
#BASE REGISTER ADDRESSING MODE:

In this addressing mode,

• Effective address of the operand is obtained by adding the content of base register with the address part of the instruction.

Effective Address = Content of Base Register + Address part of the instruction

Special purpose register that hold a base address



1. **Usage**: This addressing mode is often used in situations where a program needs to access elements of an array or a data structure located in memory. The base register typically points to the beginning of the data structure, while the offset specifies the position of a particular element within the structure.

Overall, base register addressing mode provides a flexible way to access memory locations, especially when dealing with data structures whose sizes may vary or when implementing functions with variable memory requirements.

#REAL WORLD APPLICATIONS

Addressing modes in computer organization are fundamental to various real-world applications across different domains. Here are a few examples:

1. Array Processing: Index addressing mode is extensively used in array processing tasks, such as in scientific computing, numerical simulations, and data analysis. It allows efficient access to array elements by utilizing index registers to store base addresses and displacements to access specific elements.

2. String Manipulation: String operations in programming languages often involve using addressing modes like indirect addressing or base-relative addressing. These modes enable efficient processing of strings by allowing the manipulation of individual characters or substrings within a larger string.

3. Data Structures: Addressing modes play a crucial role in implementing various data structures such as linked lists, trees, and graphs. For example, in a linked list, pointer-based addressing modes are used to traverse and manipulate the elements of the list.

4. File Systems: Addressing modes are essential in file systems for managing file storage and retrieval. They enable the operating system to access specific blocks or sectors of storage devices efficiently by using addressing modes like direct addressing or indirect addressing.

5.Graphics and Multimedia Processing: Addressing modes are utilized in graphics and multimedia applications for efficient manipulation of image and video data. Techniques like pixel addressing and block addressing rely on addressing modes to access and process image or video data at various levels of granularity.

6. Networking: In networking applications, addressing modes are used for packet routing and data transmission. Addressing modes facilitate the efficient routing of packets through networks by enabling the identification and manipulation of packet headers and addresses.

7.Embedded Systems: Addressing modes are critical in embedded systems for controlling hardware peripherals and interfacing with external devices. They allow the efficient access and manipulation of memory-mapped registers and device buffers.

Overall, addressing modes are integral to the efficient operation of computer systems across a wide range of applications, from low-level hardware interactions to high-level data processing tasks.

#RECENT TRENDS

advancements in computer architecture and system design, there hasn't been a significant shift in addressing modes themselves. However, there have been trends that influence how addressing modes are utilized and optimized within modern computer systems:

1. Vectorization and SIMD Instructions: With the increasing popularity of vectorized instructions and SIMD (Single Instruction, Multiple Data) operations in modern processors, addressing modes are often optimized to support efficient data parallelism. This involves enhancing addressing modes to facilitate vectorized memory accesses and operations on large datasets, improving performance in tasks like multimedia processing, scientific computing, and machine learning.

2. Memory Hierarchy Optimization: As memory hierarchies become more complex with the inclusion of multiple levels of cache, addressing modes are adapted to optimize data locality and cache utilization. Techniques such as cache-conscious data structures and memory access patterns influence how addressing modes are used to minimize cache misses and improve overall system performance.

3. Security and Memory Protection: Addressing modes are increasingly employed in modern processors to enhance security and memory protection mechanisms. Techniques like Address Space Layout Randomization (ASLR) and Data Execution Prevention (DEP) utilize addressing modes to randomize memory layouts and enforce restrictions on memory access, mitigating security vulnerabilities such as buffer overflows and code injection attacks.

4. Hardware Acceleration and Custom Instruction Sets: With the proliferation of specialized hardware accelerators and custom instruction sets for specific workloads, addressing modes may be tailored to optimize memory accesses and data movement within these specialized processing units. Addressing modes are customized to align with the memory access patterns and data structures commonly used in targeted applications, enhancing performance and energy efficiency.

5. Integration with Compiler Optimization: Modern compilers play a crucial role in optimizing memory accesses and addressing modes to improve program performance. Compiler optimizations such as loop unrolling, loop vectorization, and memory prefetching techniques heavily influence how addressing modes are utilized to maximize instruction-level parallelism and memory bandwidth utilization.

Overall, recent trends in addressing modes focus on optimizing memory accesses, enhancing security, and leveraging specialized hardware accelerators to improve system performance, energy efficiency, and security in modern computer architectures.