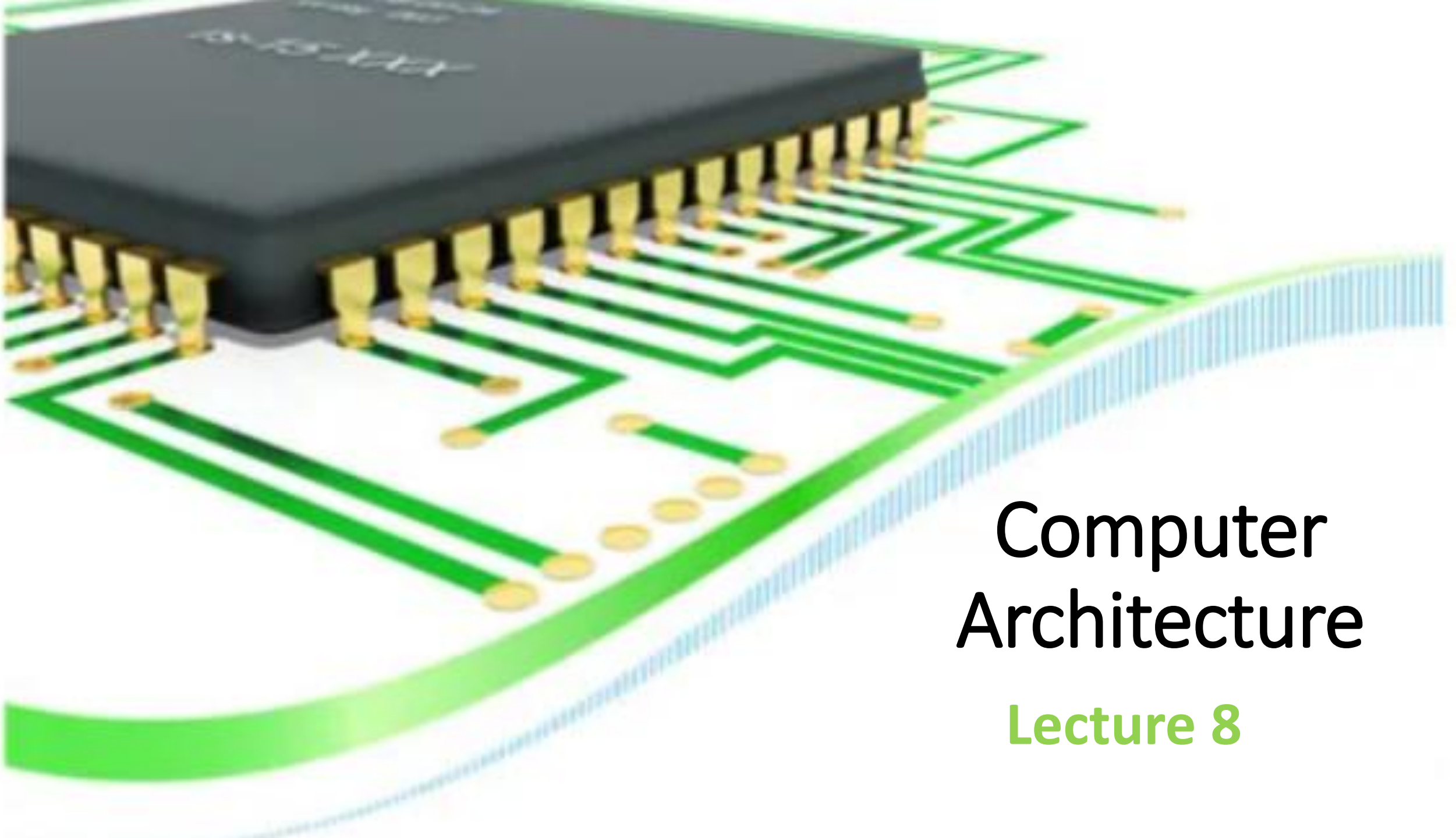


Computer Architecture

Lecture 8

Agenda

- Instruction formats
- Effective Address
- Addressing modes
- MIPS addressing modes

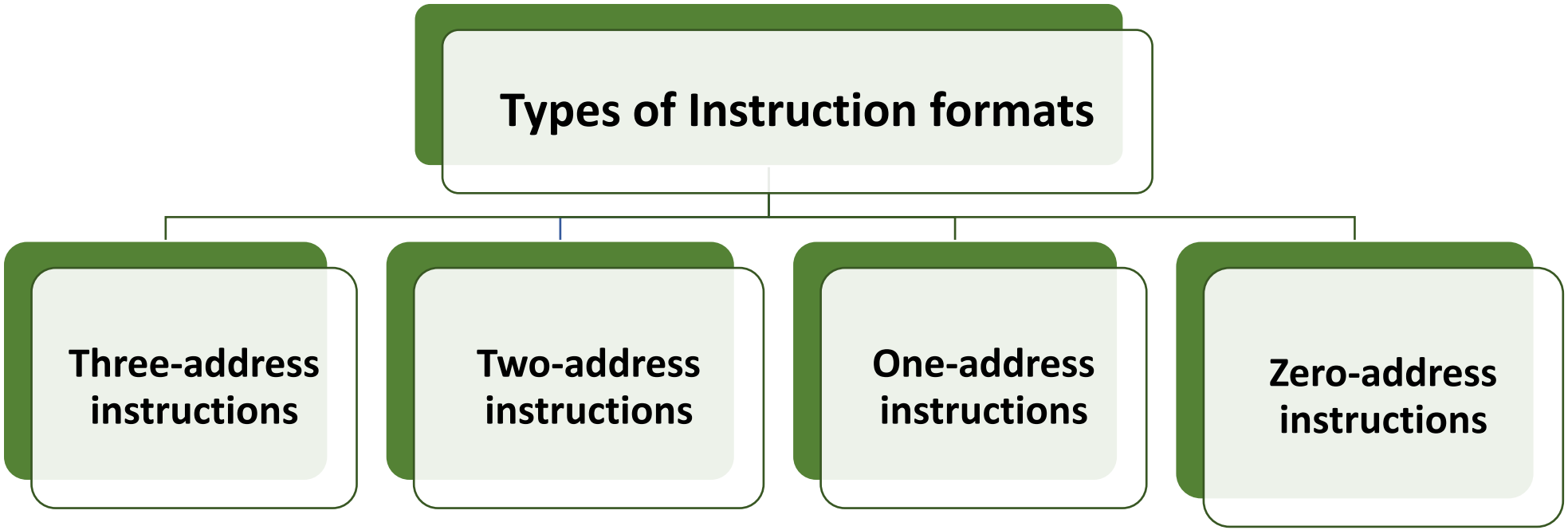
Instruction formats

- The instruction formats are a sequence of bits (0 and 1). These bits, when grouped, are known as fields. Each field of the machine provides specific information to the CPU related to the operation and location of the data.

- Instruction formats refer to the way instructions are represented in machine language.

Instruction formats

- There are several types of instruction formats, including zero, one, two, and three-address instructions.



Three-address instructions

- Three-address is a form of instructions that consists of one opcode and three fields for address. A single address field can indicate destination and two address fields are for the operand sources.

Opcode	Address 1	Address 2	Address 3

An address field could be memory address or register address

Example:

```
ADD R3, R1, R2        // R3= R1+R2
ADD C, A, B           //M[C] = M[A]+M[B]
```

Remember: A label is an address

Two-address instructions

- Two-address is a form of instructions that consists of one opcode and two fields for address. In the case of two-address instruction, the result is stored in one of the two operands.

Opcode	Address 1	Address 2

An address field could be memory address or register address

Example:

```
ADD R1, R2          //R1 = R+R2
ADD A, B            //M[A]=M[A]+M[B]
```

Remember: A label is an address

One-address instructions

- One-address is a form of instructions that consists of one opcode and one field for an address. Usually, the Accumulator register is always the first operand, it can be omitted without causing any confusion.

Opcode	Address

An address field could be memory address or register address

Example:

```
ADD R1          //AC = AC+R1
ADD B           // AC= AC + M[B]
```

Remember: A label is an address

Zero-address instructions

- Zero-address is a form of instruction that consists of an opcode. These instructions do not specify any operands or addresses. Instead, they operate on data stored in registers or memory locations implicitly defined by the instruction.

Opcode

Example:

```
ADD                 //adds the top-most elements of the stack
```

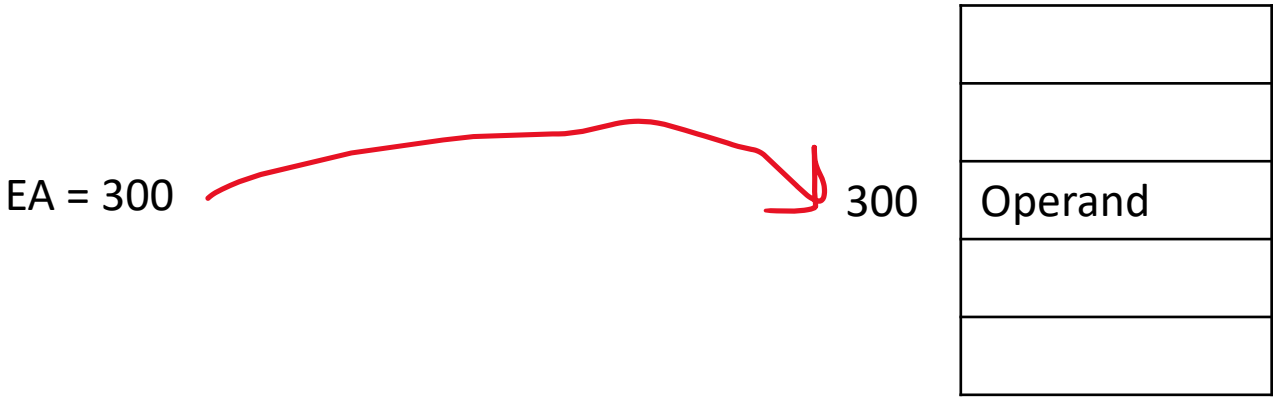
Addressing modes

- Addressing modes represent the distinct ways in which the operand of an instruction is specified with the instruction represented in binary.

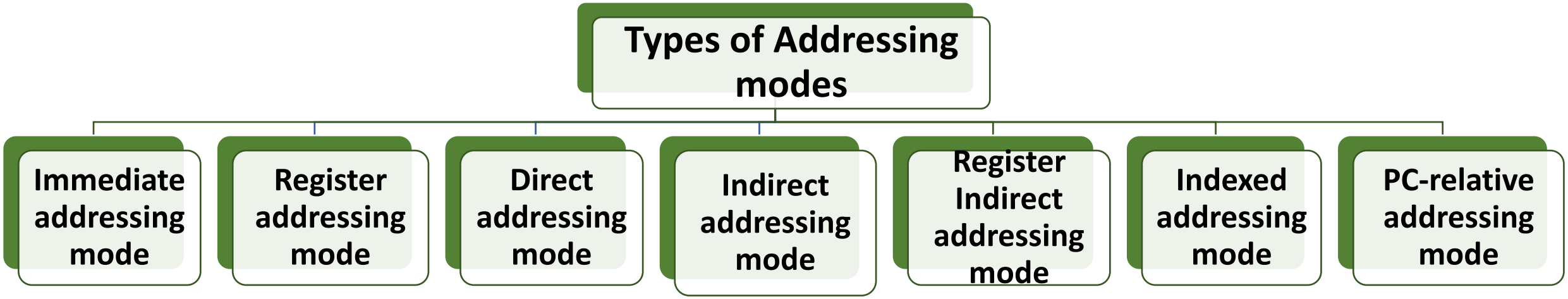
- An addressing mode specifies how to calculate the effective memory address of an operand by using information held in registers and/or constants contained within the binary representation of the instruction.

Effective Address

- An effective address is the location of an operand that is stored in memory.



Types of Addressing modes



Immediate addressing mode

- In the **immediate addressing** mode, the actual data to be used as the operand is included in the instruction itself instead of the address field.

Example:

- MOV R1, 20 //initializes register R to a constant value 20.

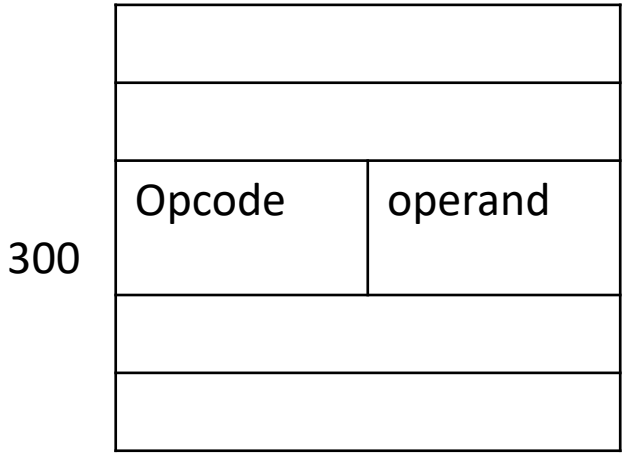
Effective address in this mode is the address of the instruction itself if the instruction is one cell.

OR

The address of the instruction immediate value field if the instruction is stored in more than one cell.

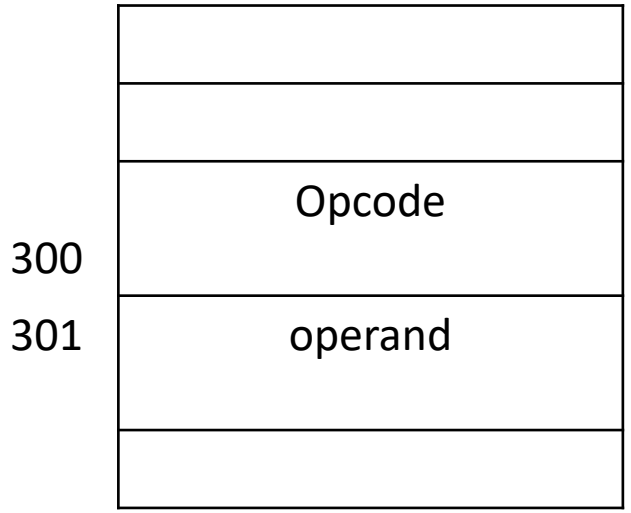
Immediate addressing mode

If the instruction is one cell.



EA = 300

If the instruction is stored in more than one cell.



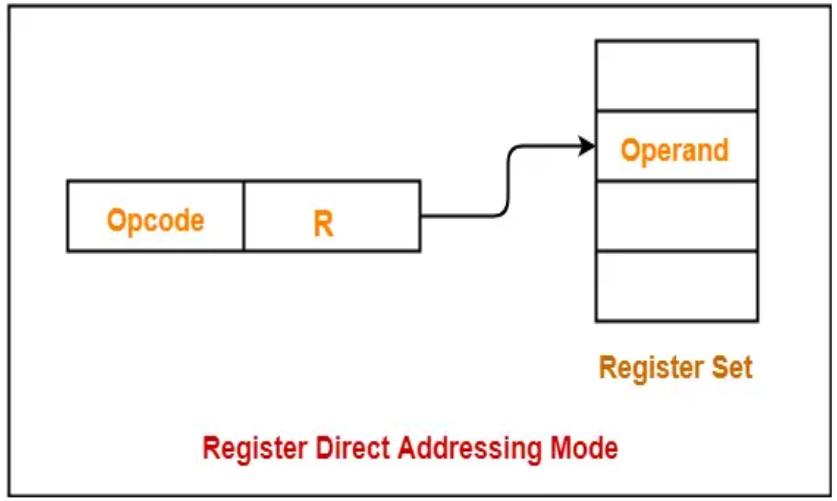
EA = 301

Register addressing mode

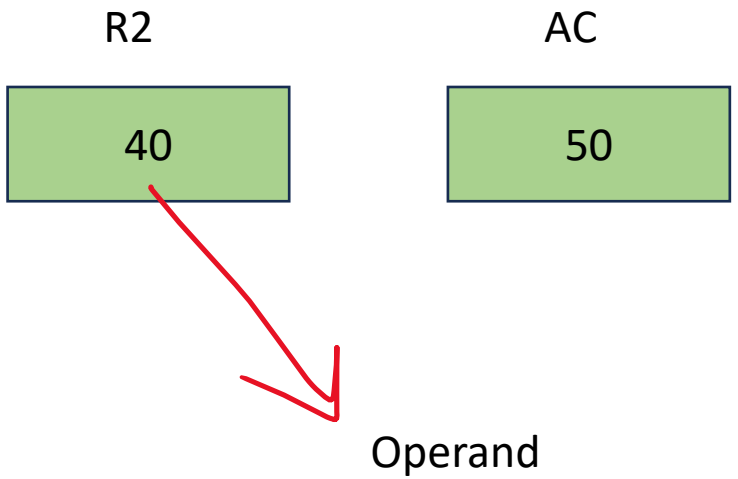
- **Register addressing** mode indicates the operand data is stored in the register itself, so the instruction contains the number of the register. The data would be retrieved from the register.

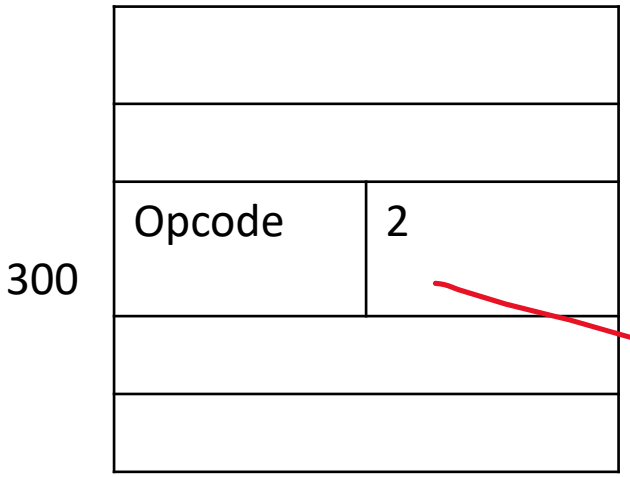
In this addressing mode,

- The operand is contained in one of the registers of the register set.
- The address field of the instruction refers to a CPU register th the operand.
- No reference to memory is required to fetch the operand.



Register addressing mode





The instruction has the number of the register having the operand

Example:

ADD R2 //will increment the value stored in the accumulator by the content of register R2.

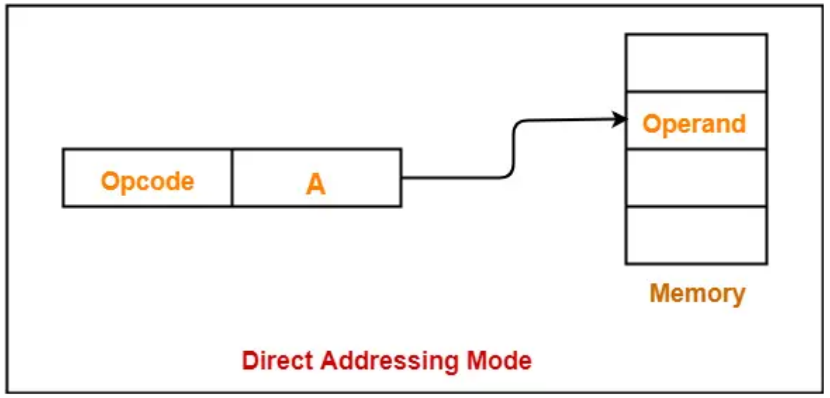
$AC \leftarrow AC + R2$

(Memory) Direct addressing mode

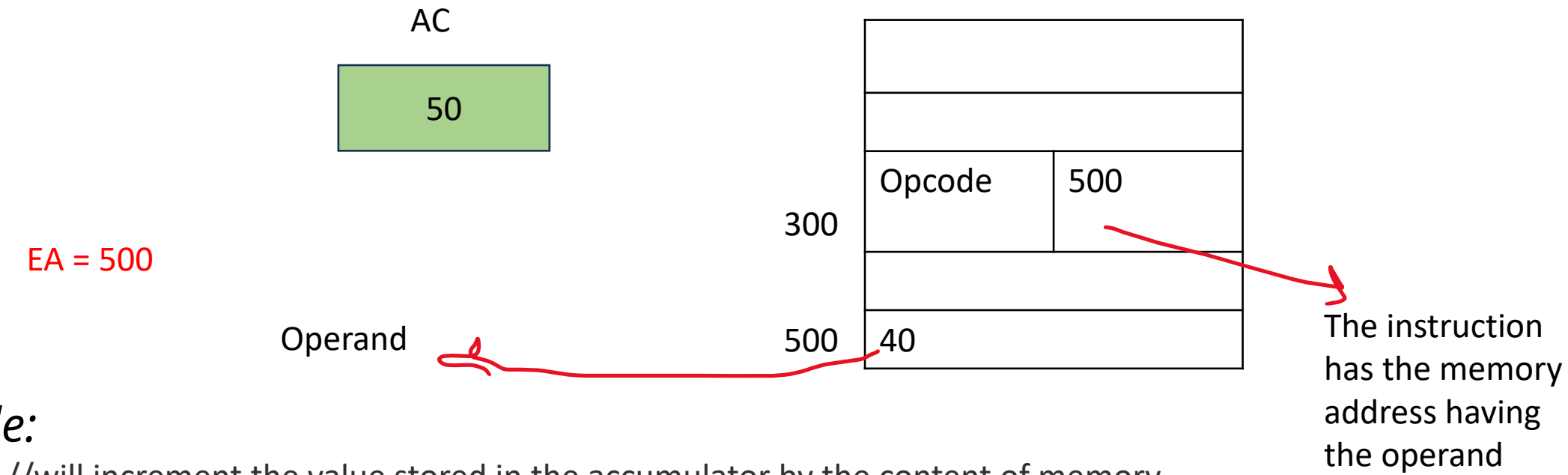
- **Direct addressing** mode indicates the operand data is stored in a memory address, and the instruction contains this memory address.

In this addressing mode:

- The address field of the instruction contains the effective address of the operand.
- Only one reference to memory is required to fetch the operand.



Direct addressing mode



Example:

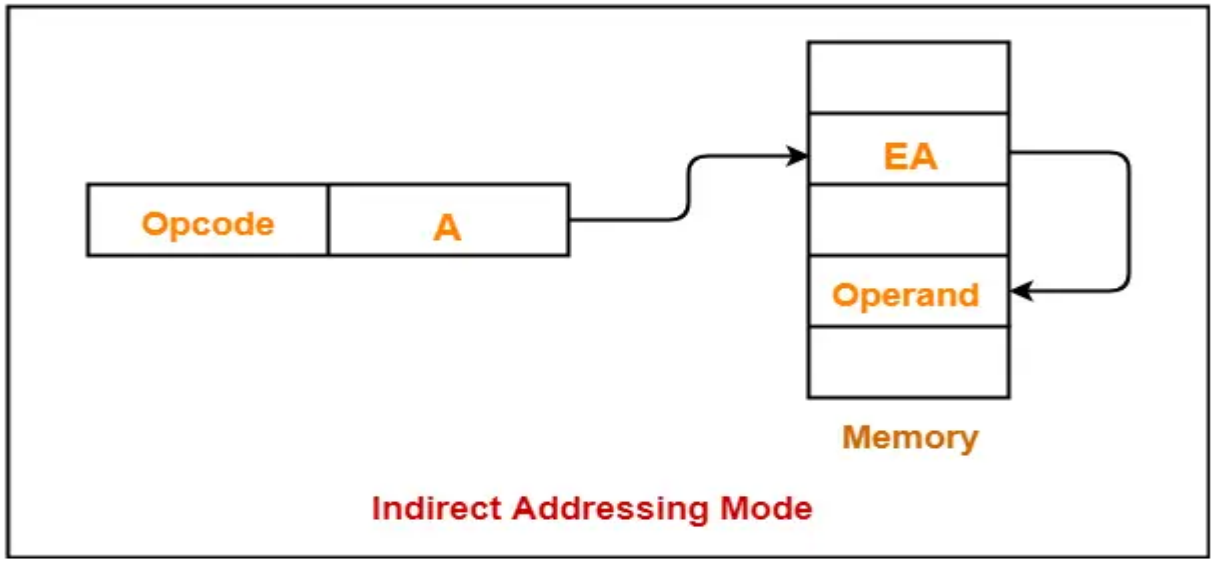
ADD 500 //will increment the value stored in the accumulator by the content of memory address 500

$$AC \leftarrow AC + M[500]$$

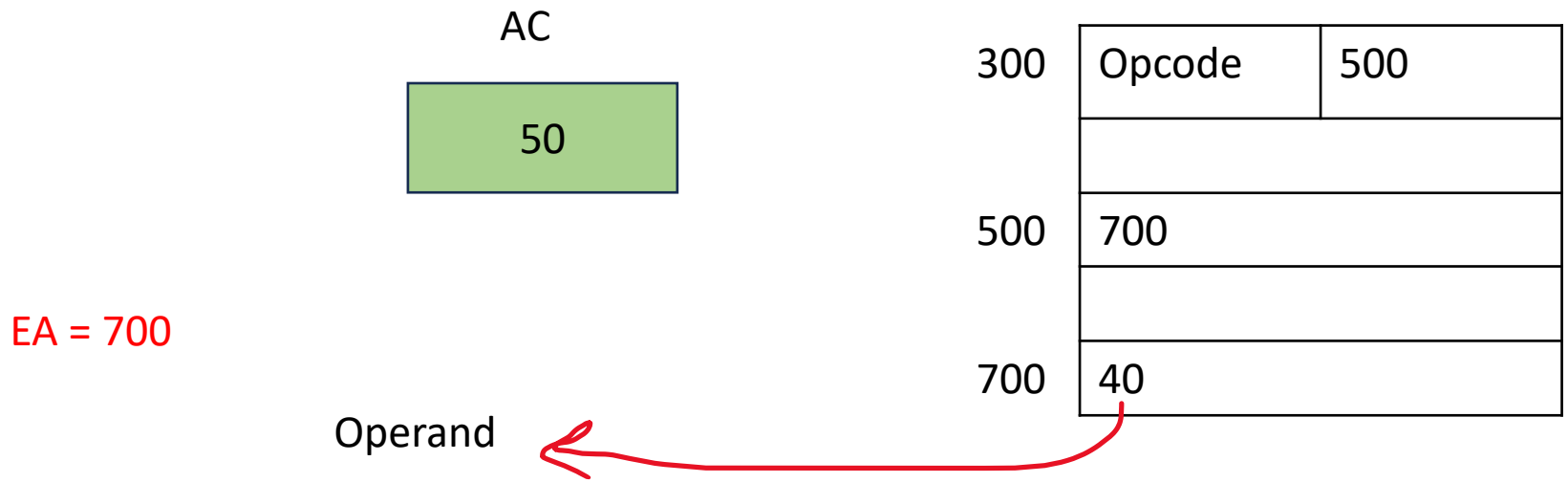
Indirect (memory) addressing mode

In this addressing mode

- The address field of the instruction specifies the address of the memory address that contains the effective address of the operand.
- Two references to memory are required to fetch the operand.



Indirect addressing mode



Example:

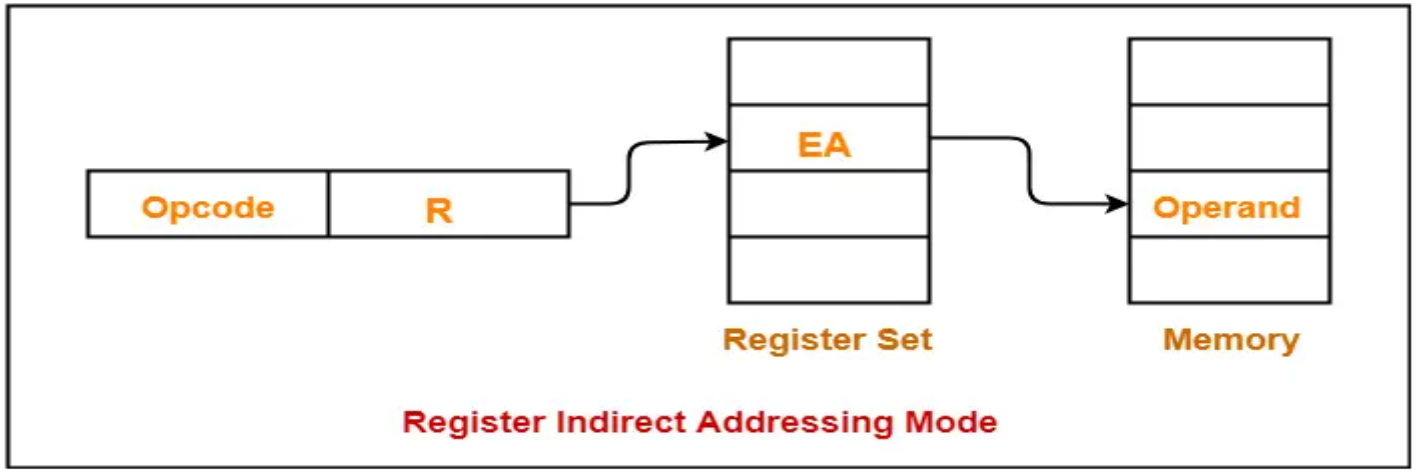
ADD 500 //will increment the value stored in the accumulator by the content of memory address 500

$$AC \leftarrow AC + M[M[500]]$$

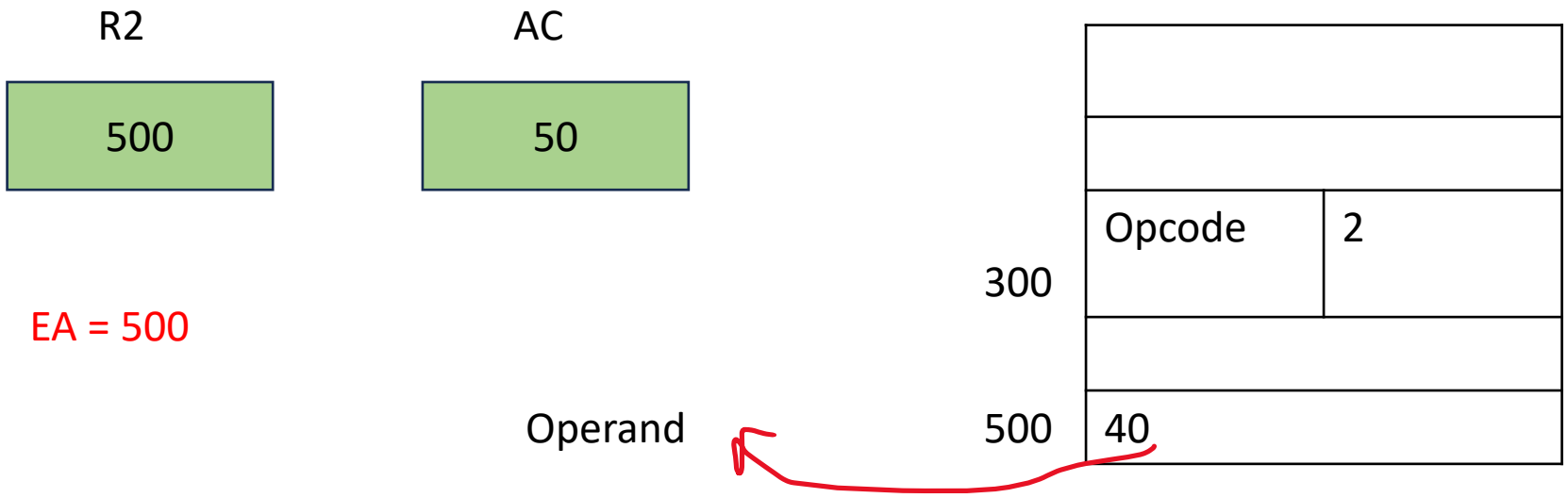
Register indirect addressing mode

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.



Register indirect addressing mode



Example:

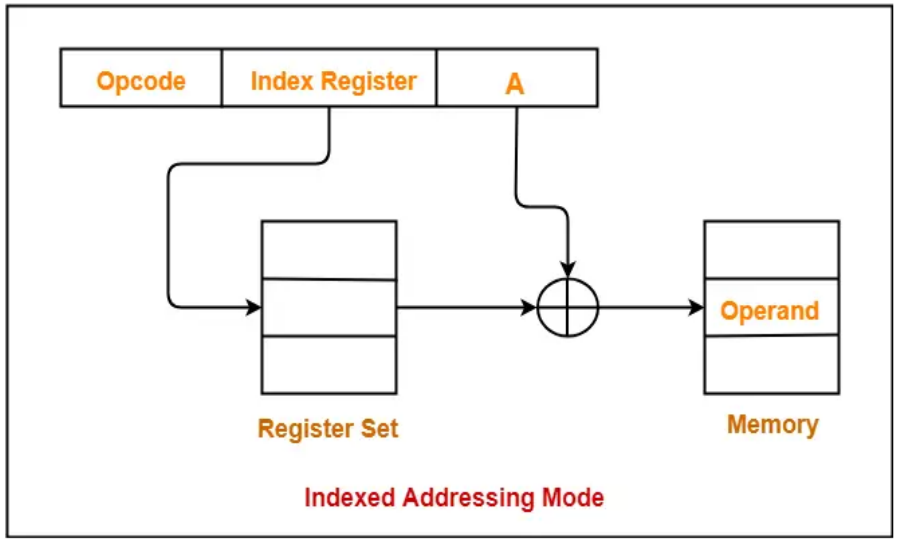
ADD R2 //will increment the value stored in the accumulator by the content of memory location specified in register R2.

Indexed addressing mode
Base addressing mode

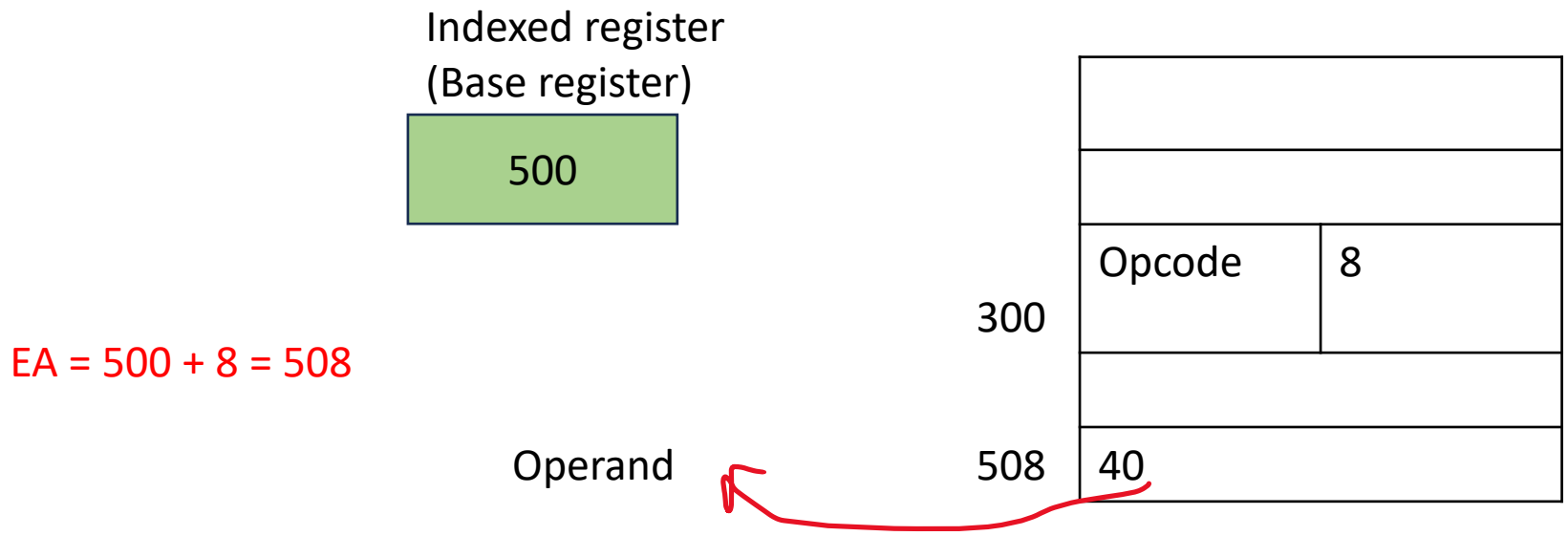
In this addressing mode,

- Effective address of the operand is obtained by adding the content of index (base) register with the address part of the instruction.

EA = Content of Index Register + offset part of the instruction



Indexed (Base) addressing mode



Example:

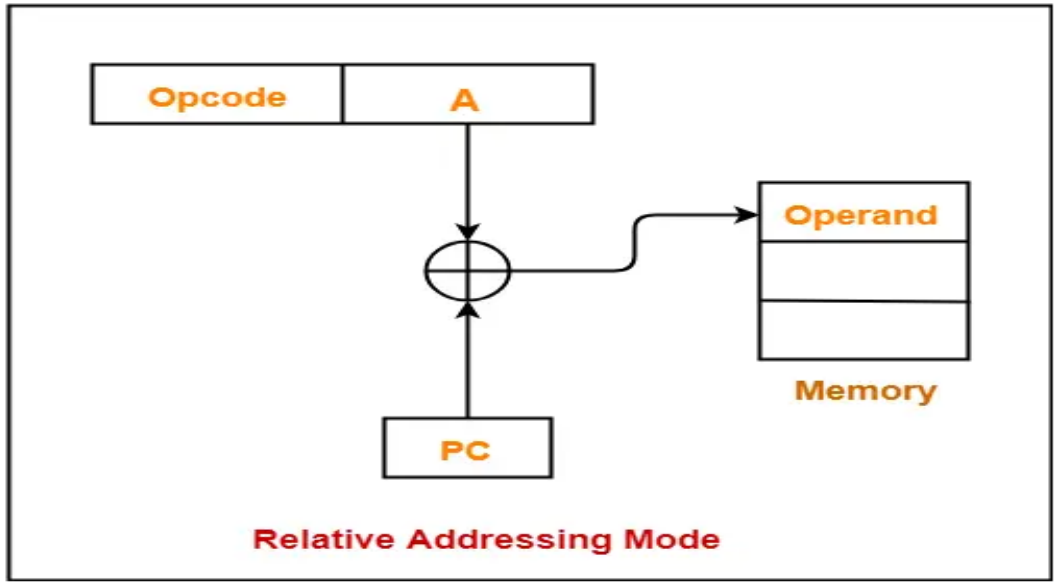
Usually used when accessing elements of an array.

PC-relative addressing mode

In this addressing mode:

- Effective address of the operand is obtained by adding the content of the program counter with the relative address part of the instruction.

EA = Content of PC +
(Relative address part of the instruction X number of cells per instruction)



PC-relative addressing mode

PC

301

300	Branch	3
304	Next instruction	

EA = 301 + 3 = 304

PC

302

EA = 302 + (3x2)
= 308

300	Branch
	3
308	Next instruction

Example:

Usually used with branch instruction

Problem 1

- Assume an architecture with a byte-addressable memory. An instruction is stored at location 700 with its address field at location 703. The address field has a value of 900. The location 900 contains the value 1140. Registers R1, R2, and R3 contain the values 1000, 2000, and 3000 respectively. What is the effective address of the instruction operand for each of the following addressing modes? Show your work.

a. Direct Memory Addressing mode.
b. Indirect memory Addressing mode.
c. Immediate
d. PC-relative
e. Register Indirect if the location 703 contains the value 2 instead of 900.
f. Indexed assuming that R3 is the index register.

Problem 1

- Assume an architecture with a byte-addressable memory. An instruction is stored at location 700 with its address field at location 703. The address field has a value of 900. The location 900 contains the value 1140. Registers R1, R2, and R3 contain the values 1000, 2000, and 3000 respectively. What is the effective address of the instruction operand for each of the following addressing modes? Show your work.

700	Opcode
703	Address = 900
704	Next instruction
900	1140

R1	R2	R3
1000	2000	3000

Solution

a. Direct Memory Addressing mode: EA = 900

b. Indirect memory Addressing mode: EA = 1140

c. Immediate: EA = 703

d. PC-relative: EA = 704 + (900x4) = 4304

e. Register Indirect if the location 703 contains the value 2 instead of 900: EA = 2000

f. Indexed assuming that R3 is the index register: EA = 3000 + 900 = 3900

700	Opcode
703	Address = 900
704	Next instruction
900	1140

R1	R2	R3
1000	2000	3000

Problem 2

- A PC-Relative mode branch type of instruction is stored in memory at an address equivalent to decimal 1000 (assume four cells per instruction). The branch is made to an address equivalent to decimal 2000. What should be the value of the relative address field of the instruction (in decimal)?

Solution:

EA = PC (new) + (relative addressx4)

2000 = 1004 + (Relative addressx4)

(Relative addressx4) = 2000 - 1004 = 996

Relative address = 996 / 4 = 249

1000	Branch
1003	???
1004	Next instruction

Problem 4

- A computer has a 32-bit instructions and 10-bit addresses. If the instructions on this computer are either of type one-address instructions or two-address instructions with an opcode field. If there are 4000 two-address instructions, how many one-address instructions can be formulated?

Solution:

Opcode	Address 1	Address 2
12-bits	10-bits	10-bits

2^{12} = 4096 different combinations.

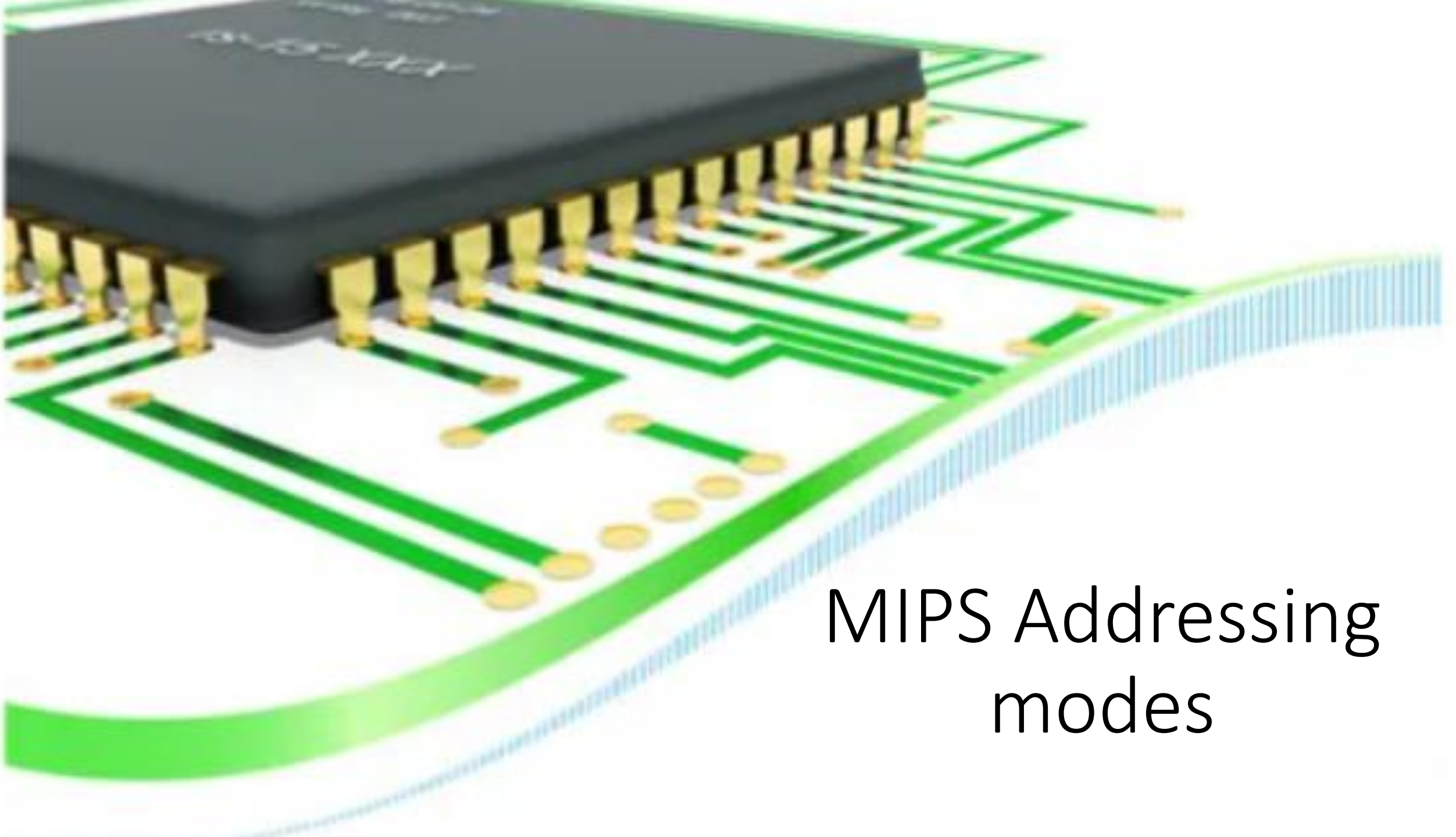
4096 – 4000 = 96 combinations can be used for one address

For one-address instructions, one of the address fields can be used as an extension to the Opcode.

Opcode	??	Address
12-bits	10-bits	10-bits

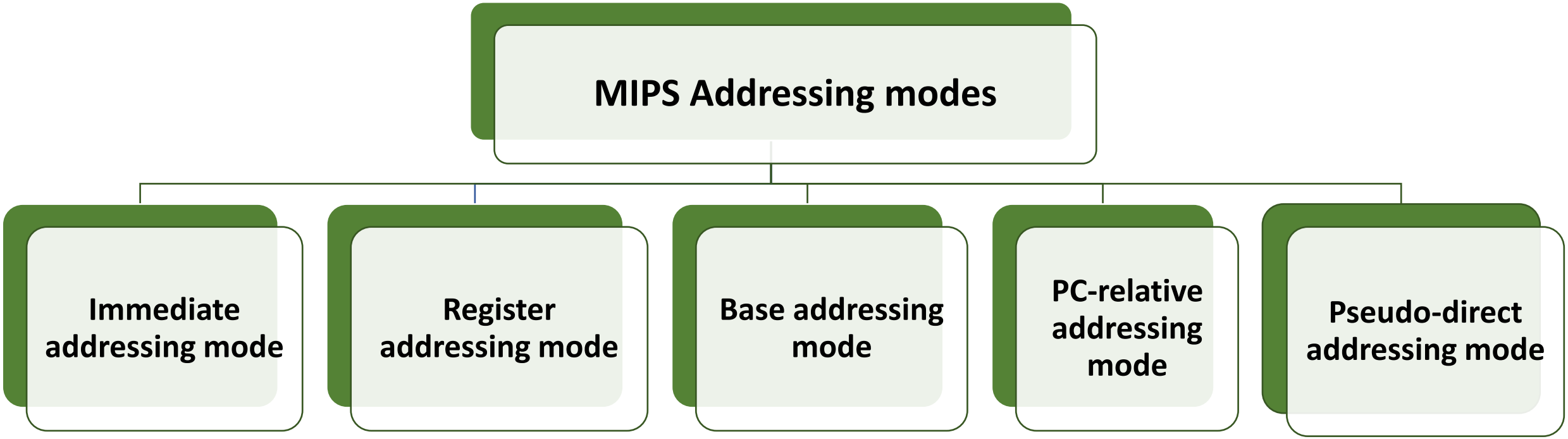
Maximum number of one address instruction = 96 x 2^{10} =

98,304 instructions



MIPS Addressing modes

MIPS Addressing modes



MIPS Addressing modes

- Immediate addressing mode:

  Example: addi $s1, 5

- Register addressing mode:

  Example: add $s1 , $s2 , $s3

- Base-addressing mode:

  Example: lw $t1 , 8 ($t2)

MIPS PC-relative addressing mode

3 instructions to the place of branch

```
200 li $s1, 5
204 li $s2, 3
208 beq $s1, $s2, label
212 add $t0, $t1, $t2
216 mul $t3, $t1, $t2
220 div $t3, $t1, $t2

224 label: sub $t0, $t1, $t2
```

beq $s1, $s2, label

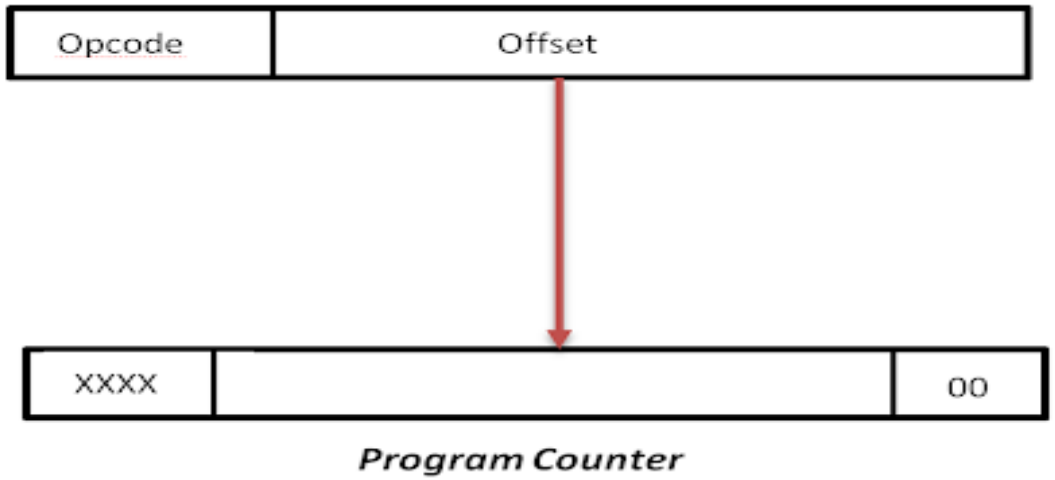
5	16	17	3

How to get the real address:
(PC+4) + (offset*4)

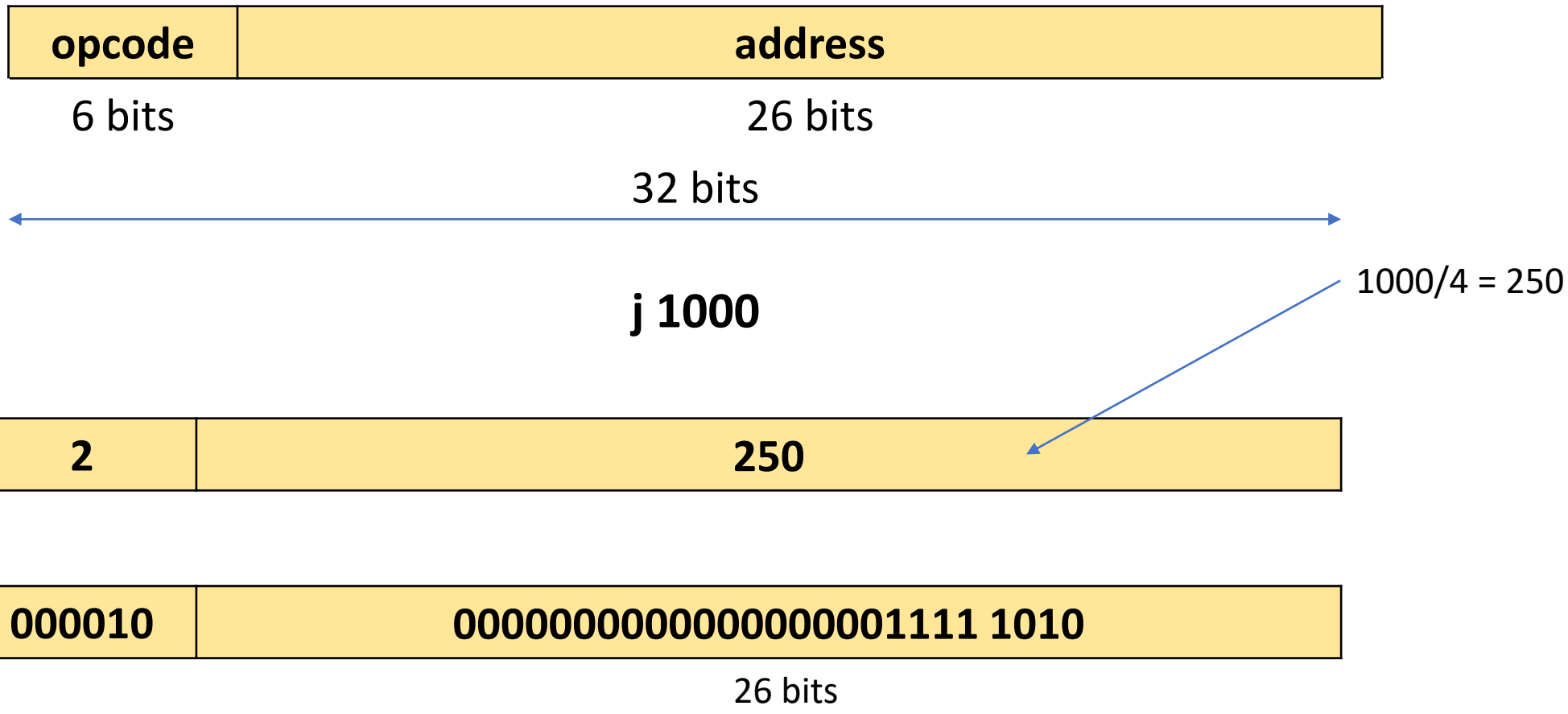
$EA = PC_{new} + (offset*4)$
$= 212 + (3*4)$
$= 212 + 12 = 224$

MIPS Pseudo-direct addressing mode

- Pseudo-Direct addressing is specifically used for J-format instructions such as jump instruction. The instruction format is 6 bits of opcode and 26 bits for the immediate value (target).
- In Pseudo-Direct addressing , the effective address is calculated by taking the upper 4 bits of the Program Counter(PC) , concatenated to the 26 bits immediate value , and the lower two bits are 00.



MIPS Pseudo-direct addressing mode



MIPS Pseudo-direct addressing mode

000010	00000000000000001111 1010

26 bits

PC

0000000001111000011111010110 1101

32 bits

Uppermost significant bits from PC

00000000000000001111 101000

28 bits

Append 00, thus the number is multiplied by 4 and in 28-bits

EA = 0000000000000000001111 101000

32 bits

References

Some images in these slides are taken from :

- https://www.gatevidyalay.com/addressing-modes/

Thank You