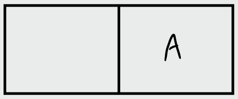
Addressing Modes

A = contents of an address field in the instruction

Immediate Addressing (MIPS uses it)

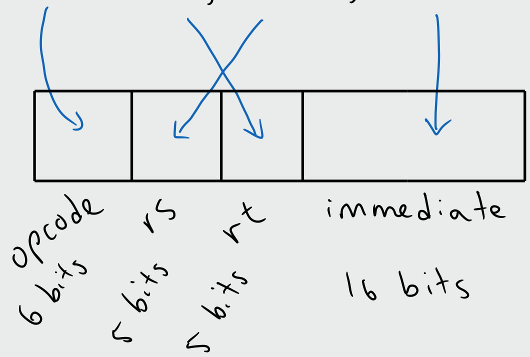


Data = A

rs: source register
rt: destination register

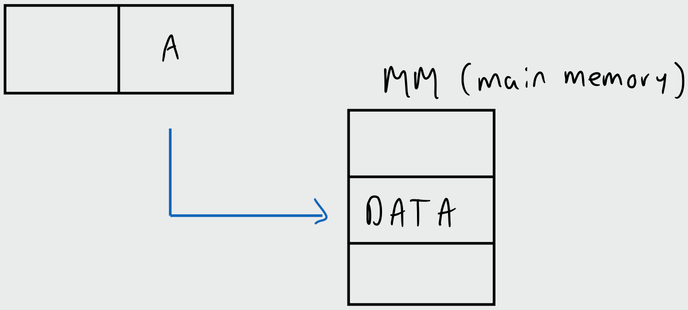
Mips: addi (add immediate)

```
addi $s0, $t1, -24      # $s0 ← $t1 - 24
```



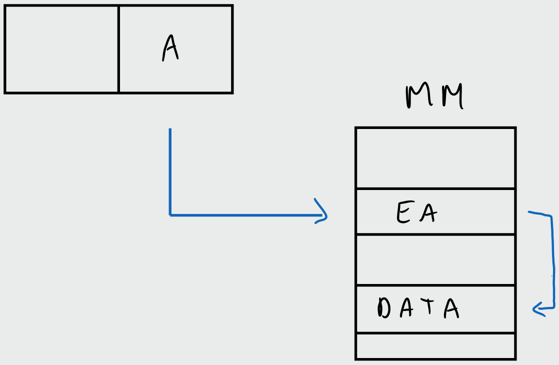
```
ori $s0, $t1, 0xAB05

$s0 ← $t1 | 0xAB05

# bitwise OR immediate
```

Direct Addressing (MIPS doesn't use it)



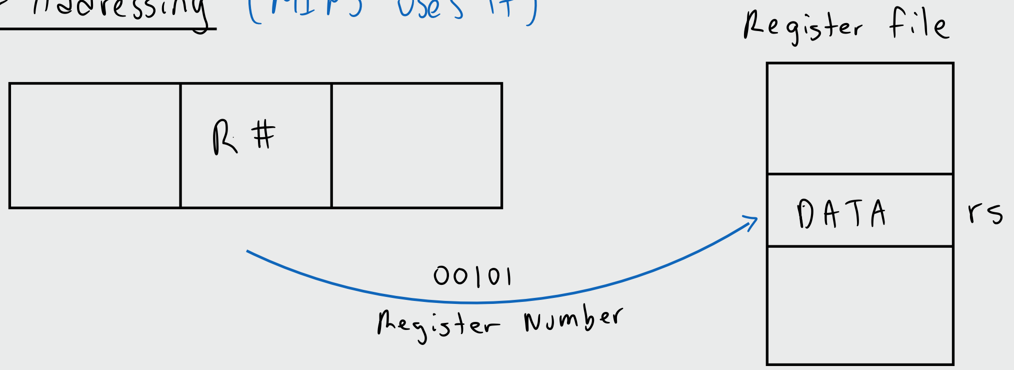
The address field A in the instruction directly specifies the memory location where the data is stored

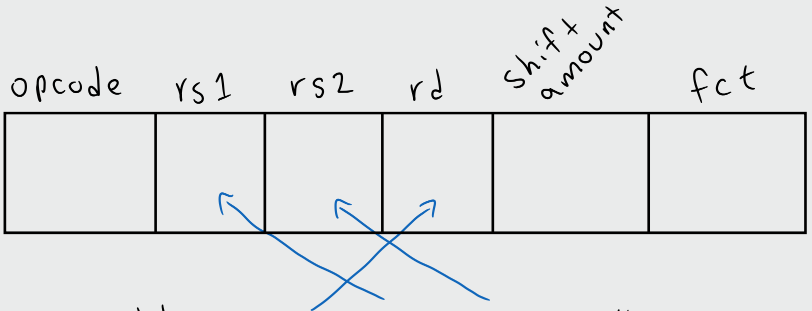
Indirect Addressing (MIPS doesn't use it)



The address field A in the instruction does not contain the actual data's memory location. Instead, A holds a memory address where the effective address (EA) is stored. The processor first retrieves the EA from this memory location, then uses it to access the actual data in main memory.

Register Addressing (MIPS uses it)



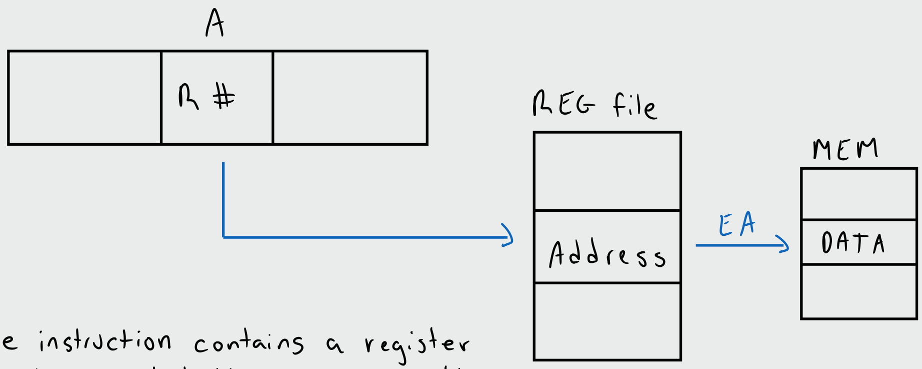


rs : first source register
rt : second souce register
rd : destination register

add $s1, $t1, $t0 # $s1 ← $t1 + $t0

- the instruction contains a Register Number
- the processor retrieves the data from the specified register in the register file
- this eliminates the need to access memory, improving speed

REG Indirect Addressing (MIPS doesn't use it)



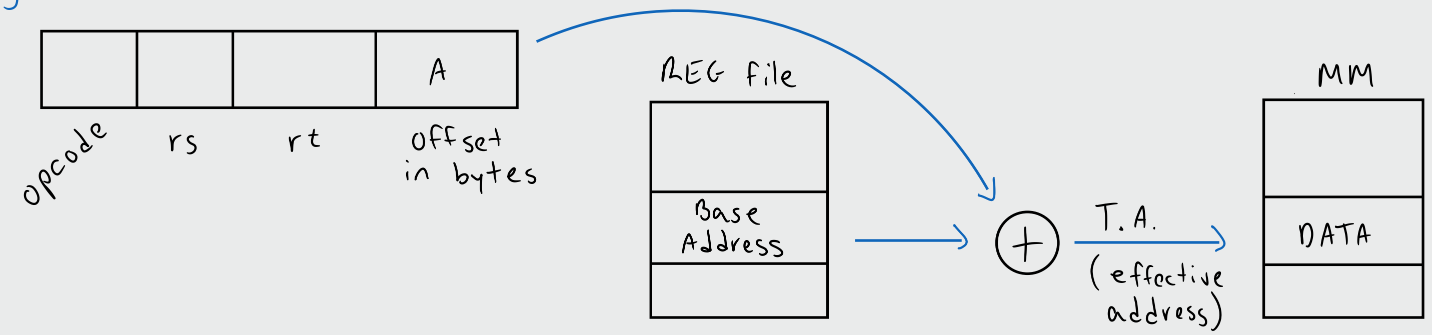
- The instruction contains a register number that holds a memory address.
- The processor fetches the effective address from the specified address in the register file.
- The processor uses this EA to access memory and retrieve data

Base-Register Addressing (MIPS uses it) (Displacement Addressing)

Used for lw and sw

T.A. = Base Address + offset (target address)

eg: lw $t1, 24($s0) # load



- The base register holds a memory address
- The offset is a signed immediate value included in the instruction
- the processor computes the target address (TA) by adding the offset to the value in the base register
- The TA is then used to access memory

Example

Consider an instruction. the address field of the instruction contains the value 2000.

When needed, register #18 is used. Register 18 contains the value 1600

The list below shows a few addresses and the memory content of each of those addresses.

Address (bytes)	Memory Content
48	844
2000	3000
1600	400
2500	800
3000	1200
3600	500

=>

Addressing Mode	Effective Address (bytes)	Value (DATA)
Immediate	—	2000
Direct	2000	3000
Indirect	3000	1200
Register	REG #18	1600
Register Indirect	1600	400
Displacement	3600	500

We'll refer to the address field of the instruction as A

- Immediate: A contains 2000, grab it immediately
- Direct: A contains 2000, an address of a mem location that contains our data => 3000
- Indirect: A contains 2000, an address of a mem location that contains an address (3000) of a memory location that contains our data => 1200
- Register: Uses the register number given in the instruction (18) which contains our data => 1600
- Register Indirect: Uses the register number given in the instruction (18) which contains an address (1600) of a memory location that contains our data => 400
- Displacement: EA = (Value in base register) + (offset from instruction)
  = Value in base register (18) + offset (2000)
  = 1600 + 2000 = 3600 => ADDR 3600 contains our data => 500