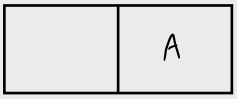
Addressing Modes

A = contents of an address field in the instruction

Immediate Addressing (MIPS uses it)

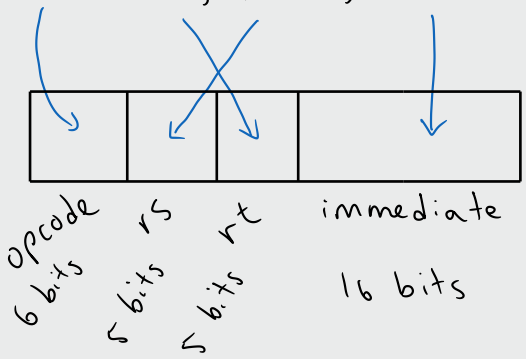


Data = A

rs: source register
rt: destination register

Mips: addi (add immediate)

addi $s0, $t1, -24 # $s0 ← $t1 - 24

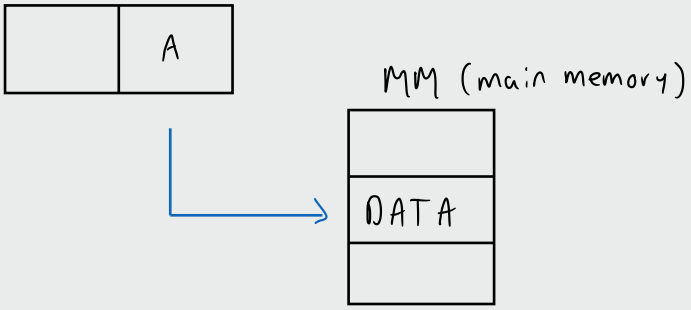


ori $s0, $t1, 0xAB05

$s0 ← $t1 | 0xAB05

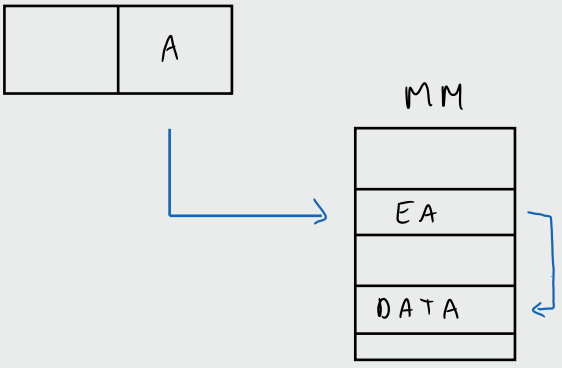
bitwise OR immediate

Direct Addressing (MIPS doesn't use it)



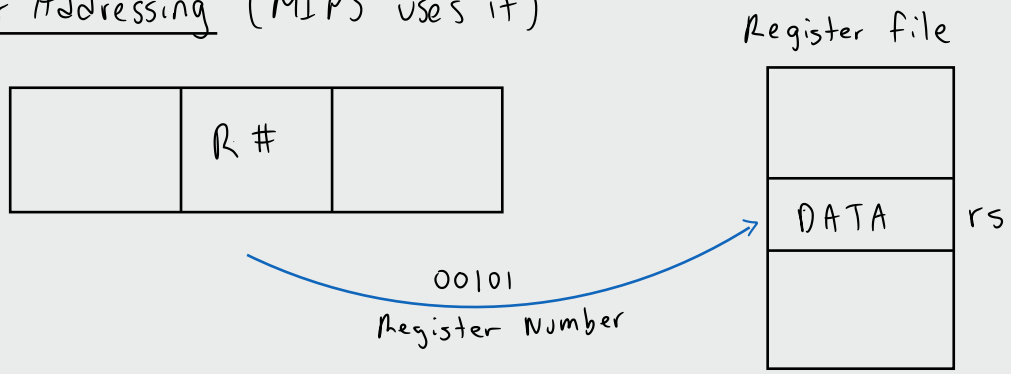
The address field A in the instruction directly specifies the memory location where the data is stored

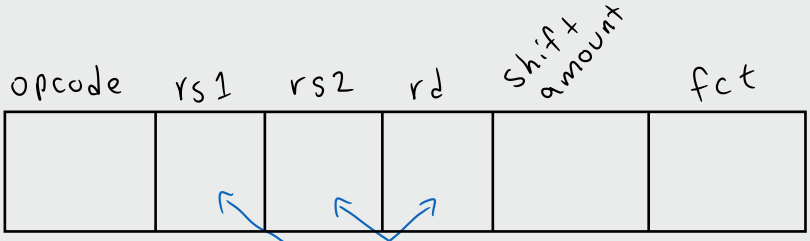
Indirect Addressing (MIPS doosn't use it)



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)



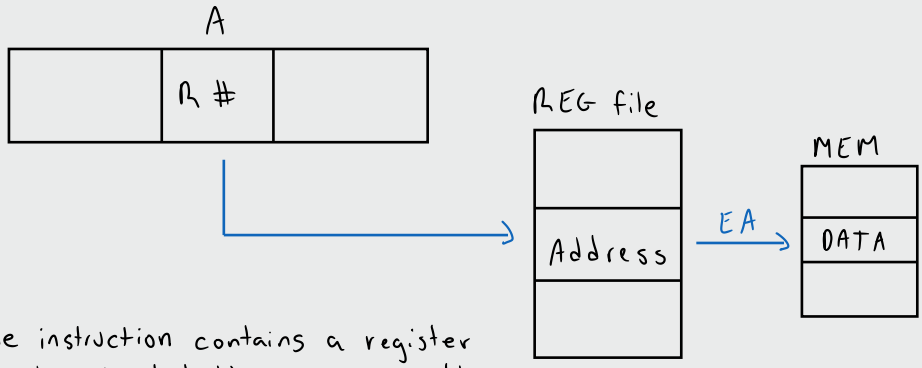


rs: first source register
rt: second source register
rd: destination register

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)

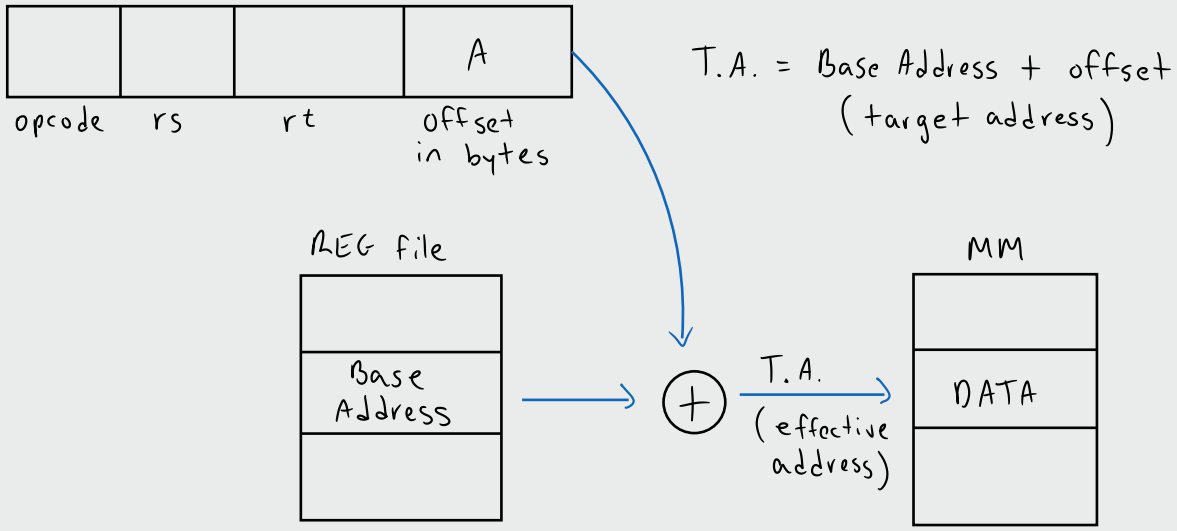


- The instruction contains a register number that holds a memory address.
- The processor fetches the effective address from the specified address in the register file.
- The processor uses this EA to access memory and retrieve data

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

We can use lw and sw

```
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