MIPS Processors

These were used primarily in the 1980s and 1990s as Unix workstations (Silicon Graphics, Decstations, etc.), even the PS2.

- Now primarily used in embedded systems

Why MIPS?

- Easiest to learn
- Covered in many textbooks
- Similar to other important architectures

Words

Main memory is organized in words.

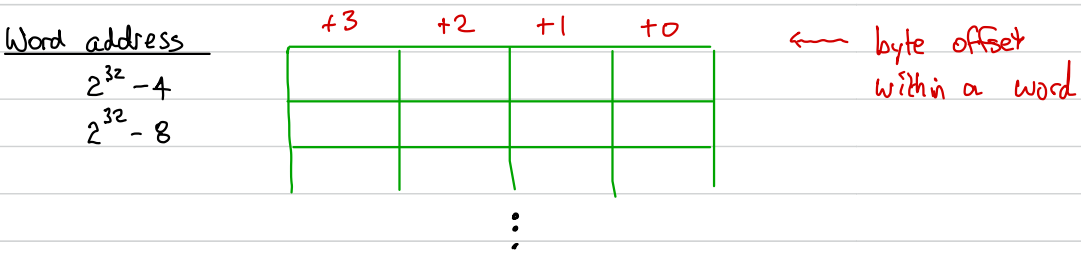
- C-programmer's view of memory: giant array of bytes
- Hardware designer / assembly-language programmer's view: giant array of words.

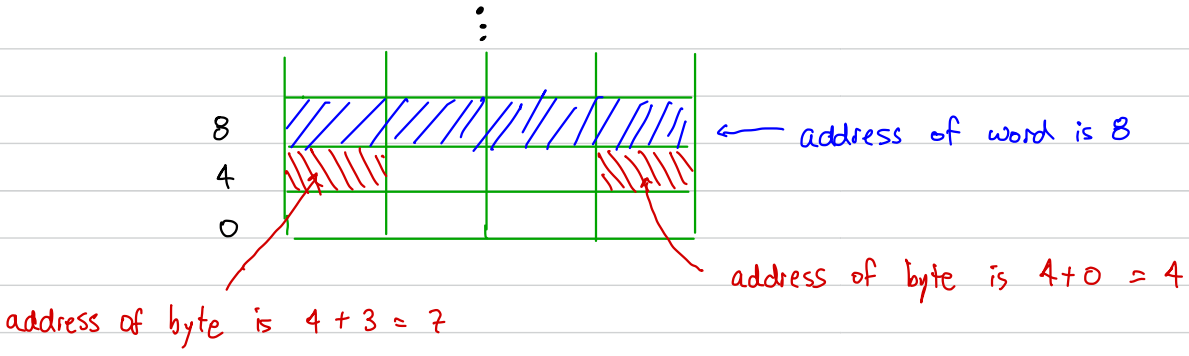
Word: a grouping of several bytes

Word size	System
2 bytes	early PCs and Macs, lower-power embedded systems
4 bytes	newer PCs and Macs
8 bytes	servers, newest game consoles, modern PCs, Macs

MIPS memory organization

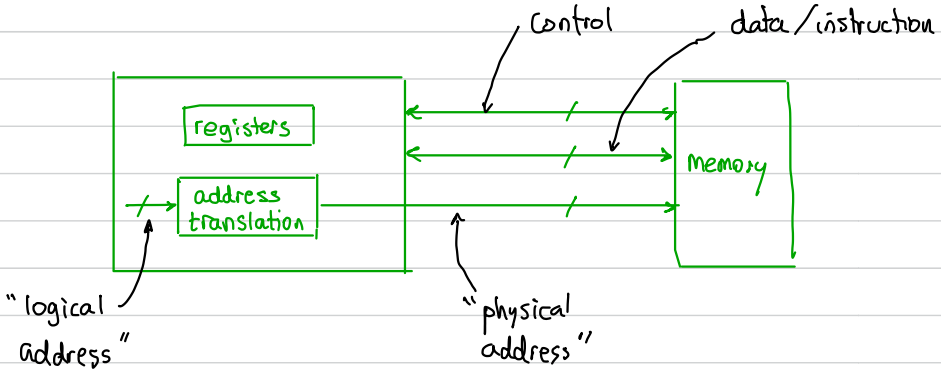
Memory is an array of 32-bit words





Note the order : MIPS convention is to list addresses with the highest at the top.

Memory access in a MIPS processor



Logical address → addresses seen by the programmer
Physical address → real address accessed in memory

} coming back to this in March!

Data access and important alignment rules

- Memory words must have addresses that are multiples of 4 (e.g., 4, 8, 0x1000, etc.)
- Memory half-words (2 bytes) must be multiples of 2.

MIPS registers

- 32 General Purpose Registers (GPRs) – 32 bits each – designated $0 - $31
  - can hold C-language ints or pointers
- 16 Floating-Point Registers (FPRs) – 64 bits each
  - can hold C-language doubles

- Some special-purpose registers

MIPS instructions

These are exactly one word (4 bytes) in size.

The Program Counter

This is one of the special-purpose registers
- used to hold the address of the next instruction

Typical instruction cycle:

repeat
1. Read instruction at the address in PC, add 4 to the PC
2. Execute instruction just read

MIPS instruction example

Typical add instruction

```
add   $17, $8, $9
```

$17: result goes to GPR 17

$8, $9: add these two GPRs

These three fields tell the processor to add two registers, place the result in some other register

Useful notation for long numbers

Grouping digits by whatever group size is convenient

E.g., 1 billion (base 10) 1_000_000_000 ← underscores

$2^{32} - 1$ in hex ffff_ffff