A processor's registers

These are storage locations for bytes that are inside the processor core

- not part of main memory
- not very many of them

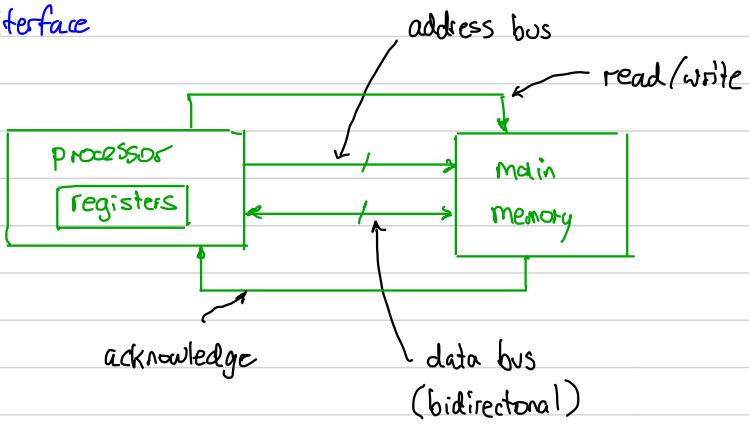
Example:

CPU	No. of general-purpose registers	size in bytes
x86	8	4
x86_64	16	8
MIPS-32	32	4

Processor reads and writes to main memory

Needed to transfer bytes of data between the registers and main memory.

Simplified memory interface



Memory reads:

Read sequence:

1. processor sets address
2. processor commands memory to read
3. memory gives data to processor
4. memory acknowledges "done".

Write sequence:

1. processor sets address
2. processor gives data to write
3. processor commands memory to write
4. memory acknowledges "done."

Usually, groups of bytes (1, 2, 4, or 8) are transferred on the data bus

- if more than 1 byte is transferred, memory bytes are accessed at sequential memory addresses starting at the address of the first byte.

Simplistic overview of how a computer works.

Two steps:

Step 1: ⟶ processor reads one or more bytes from memory (these form a machine instruction)

Step 2: ⟶ processor executes the instruction (i.e., it performs a very specific task based on the instruction bits)

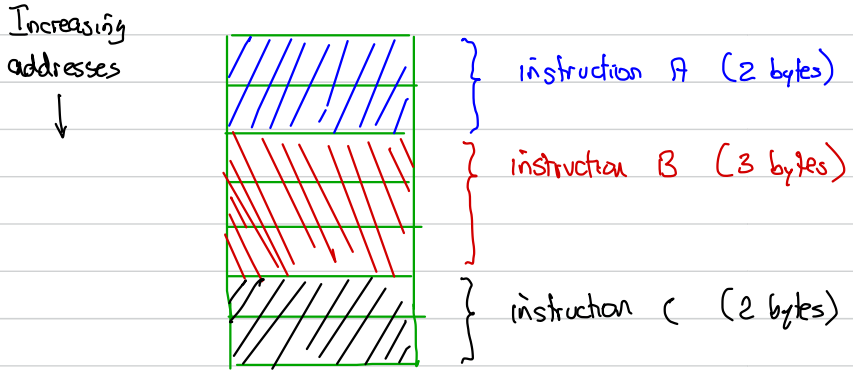
e.g.,
- add the contents of two registers
- load the contents of a register from main memory.

Sequence: 1 ⟶ 2 ⟶ 1 ⟶ 2 ⟶ ....

Sequence of instructions:

Normally, step 1 above accesses instructions in sequence, exactly as stored in memory

Sample:



Processor sequence:
Read A (both bytes)
execute A
Read B (all 3 bytes)
execute B

Read C (both bytes)
execute C

Instructions are read/executed sequentially like this until special branch or jump instructions are encountered.

These special instructions are executed when we have

if/else → requires forward branch to skip instructions

loop (while, for) → requires a backward branch to repeat some instructions

function call → requires a jump to the first instruction of the function

Using registers and memory

The registers in a processor are often called general-purpose registers (GPRs)

- GPRs hold a small amount of a program's data
- Memory holds a program's instructions and any large amounts of data (like arrays)

Execution of a C-language statement - example

Suppose:
```
int  x, y, z, foo;
foo = x + y + z;
```

Possible sequence of instructions:

1. Read 'x' from memory into GPR 1.
2. Read 'y' from memory into GPR 2.
3. Add GPR2 to contents of GPR 1.
4. Read 'z' into GPR 3
5. Add GPR3 to GPR 1
6. Write GPR1 to memory allocated for 'foo'.