Hardware (as exposed to software)

- $1+31 = 32$ registers (register 0 always contains 0)
- $2^{30}$ words of main memory (1 word = 4 bytes = 32 bits)
  - $2^{32}$ bytes of main memory
- a program counter (PC)
  - “points to” the next instruction to execute

Instruction Set (expressed in assembler)

- add rd, rs, rt  
  Example: add $8, $4, $5
- sub rd, rs, rt  
  sub $8, $5, $6
- addi rt, rs, immed  
  addi $8, $4, 100
- lw rd, offset(base)  
  lw $8, 20($4)
- sw rt, offset(base)  
  sw $8, 24($5)

Instruction Encoding (binary – what’s stored in memory)

- add (R-format)

```
 000000  | 26 25 21 20 16 15 11 10  6  5  0
---+-------------------
     | rd  rt  rs
     +-------------------
     | ADD  SPECIAL
     +-------------------
```

- addi (I-format)

```
 001000 | 26 25 21 20 16 15 11 10  6  5  0  immediate
---+-----------------------------------
     | rs  rt
     +-----------------------------------
```

- lw (I-format)

```
100011 | 26 25 21 20 16 15 11 10  6  5  0  offset
---+-----------------------------------
     | base rt
     +-----------------------------------
```

Compiling

- Going from a high-level language representation of a program to an equivalent sequence of assembler instructions