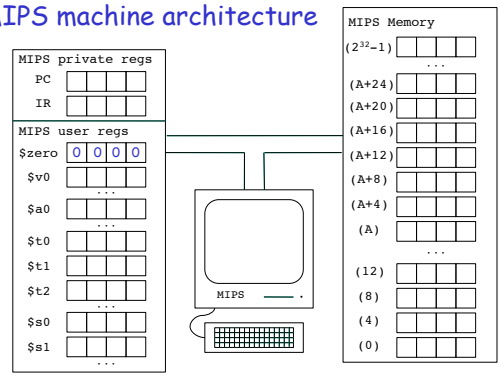
Design Principles

Conventional machine level architecture

CS240 Computer Organization
Department of Computer Science
Wellesley College

MIPS machine architecture



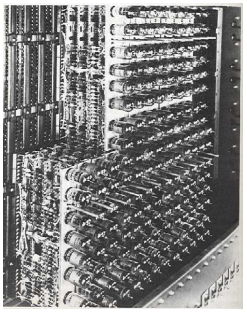
Design Principle 1. Simplicity favors regularity

Category	Instruction	Example	Meaning
Arithmetic	add	add $s1,$s2,$s3	$s1=$s2+$s3
	subtract	sub $s1,$s2,$s3	$s1=$s2-$s3

*Always three register operands.

Design Principle 2. Smaller is faster

- Operands are restricted to a small number of special locations, known as *registers*, built directly in hardware.
- The size of a MIPS register, known as its *word size*, is 32 bits.
- There are typically 32 32-bit registers in a MIPS machine.



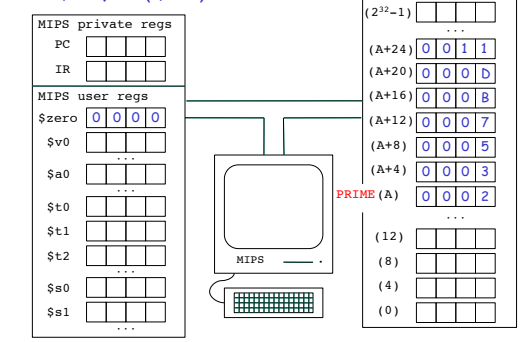
Here are some of them

Name	Register Number	Usage
$zero	0	the constant value 0
$v0-$v1	2-3	values for results and expression evaluation
$a0-$a3	4-7	arguments
$t0-$t7	8-15	temporaries
$s0-$s7	16-23	saved
$t8-$t9	24-25	more temporaries
$gp	28	global pointer
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	return address

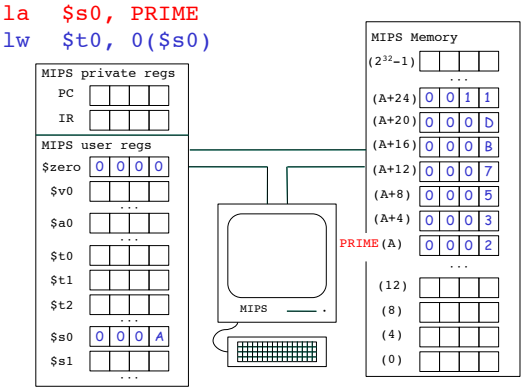
What happens when we run out of room?

- Arithmetic operations occur only on CPU registers.
- When those registers are used up, then MIPS must transfer information between memory locations and the CPU.

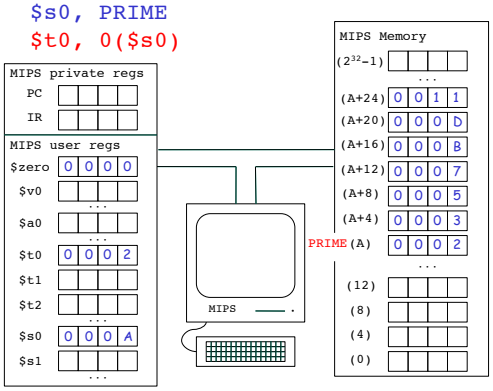

```
la   $s0, PRIME
lw   $t0, 0($s0)
```




```
la   $s0, PRIME
lw   $t0, 0($s0)
```

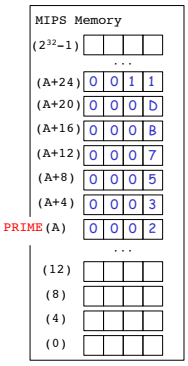


```
la   $s0, PRIME
lw   $t0, 0($s0)
```



Why the indirection?

- Memory is a large single-dimensional array of bytes with the address acting as the index to a give byte location.
- This makes array implementation a piece of cake.
- Suppose PRIME is an array of the first 100 prime numbers. How do we access the fifth prime number.

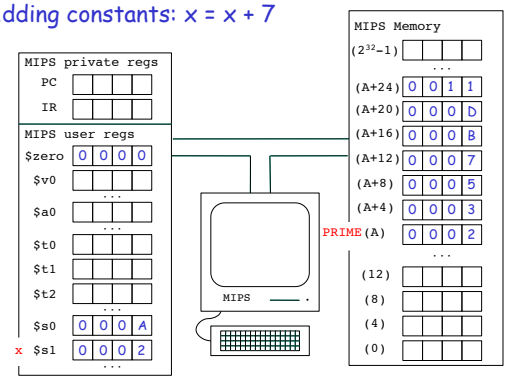


A growing vocabulary

Category	Instruction	Example	Meaning
Arithmetic	add subtract	add $s1,$s2,$s3 sub $s1,$s2,$s3	$s1=$s2+$s3 $s1=$s2-$s3
Data transfer	load word store word	lw $s1,100($s2) sw $s1, 100($s2)	$s1=Mem[$s2+100] Mem[$s2+100]=$s1

Puzzler: Add the third and fourth primes stored the array PRIME and store the result in the word location immediately preceding PRIME.

Adding constants: x = x + 7



Design Principle 3.
Make the common case fast

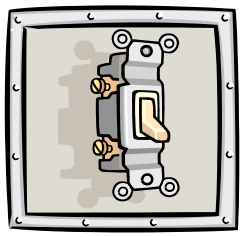
Category	Instruction	Example	Meaning

Arithmetic	add
	subtract
	add immediate
		add $s1,$s2,$s3
		sub $s1,$s2,$s3
		addi $s1,$s1,7
			$s1=$s2+$s3
			$s1=$s2-$s3
			$s1=$s1+7

Data transfer	load word store word	lw $s1,100($s2) sw $s1, 100($s2)	$s1=Mem[$s2+100] Mem[$s2+100]=$s1

Bits, bytes, and nibbles

- The basic unit of storage is a Binary digIT, or bit. Bits are organized into registers, and registers into memory arrays.
- When these registers are a part of main memory we will refer to them as locations.
- Data and programs are stored as a series of high and low electronic signals.



Positional representation

- The value represented by a digit in a positional representation system is the digit's own value multiplied by a power of the system's base.

1	4	9	2	digits
10^3	10^2	10^1	10^0	positional value in base 10

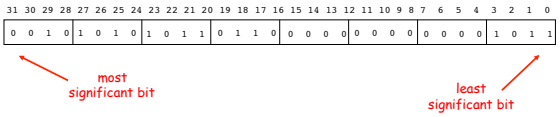
1	0	1	1	digits
2^3	2^2	2^1	2^0	positional value in base 2

Bose Hexe

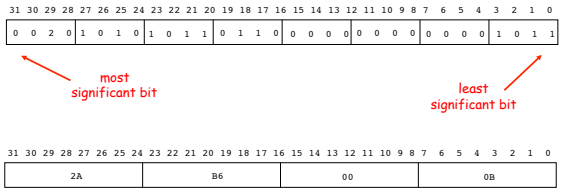
- Reading and writing binary is long and tedious, but computers eat it up.
- Computer scientists prefer hexadecimal.*
- The Romans ignored positional representation altogether.

*The rest of the world thinks we're nuts.

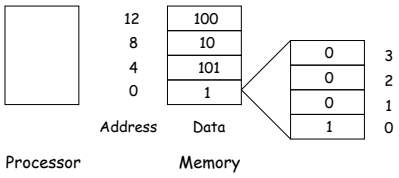
MIPS word size



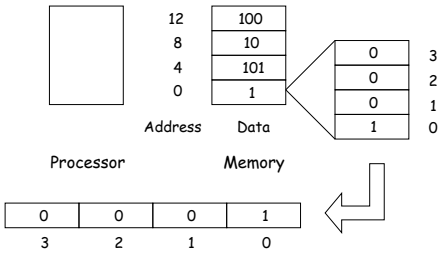
In hex



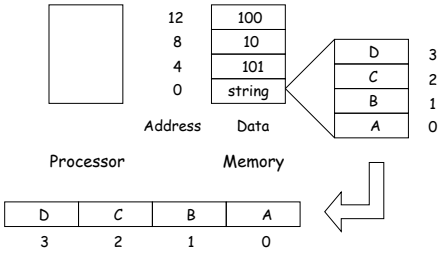
Big-endian or little-endian?



Byte order on a little-endian machine

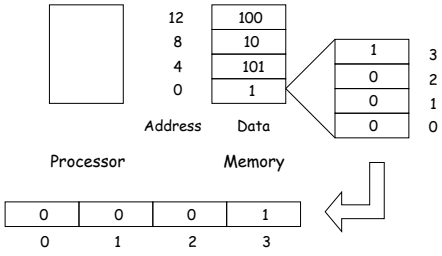


Strings appear wonky in little-endian byte order

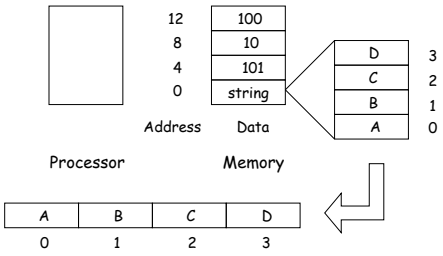


Byte order on a big-endian machine

Numbers look funny written vertically.



But big endian works better for strings



Storage Allocation Example

Sketch memory for the following MIPS program.*

```
        .data
    str:.byte       1,2,3,4
        .half       5,6,7,8
        .word       9,10,11,12
        .space      5
        .word       9,10,11,12
        .asciiz     “ABCD”
        .ascii      “ABCD”
        .byte -1

        .text
        li          $v0,10
        syscall
```

*Careful, endian-ness counts.