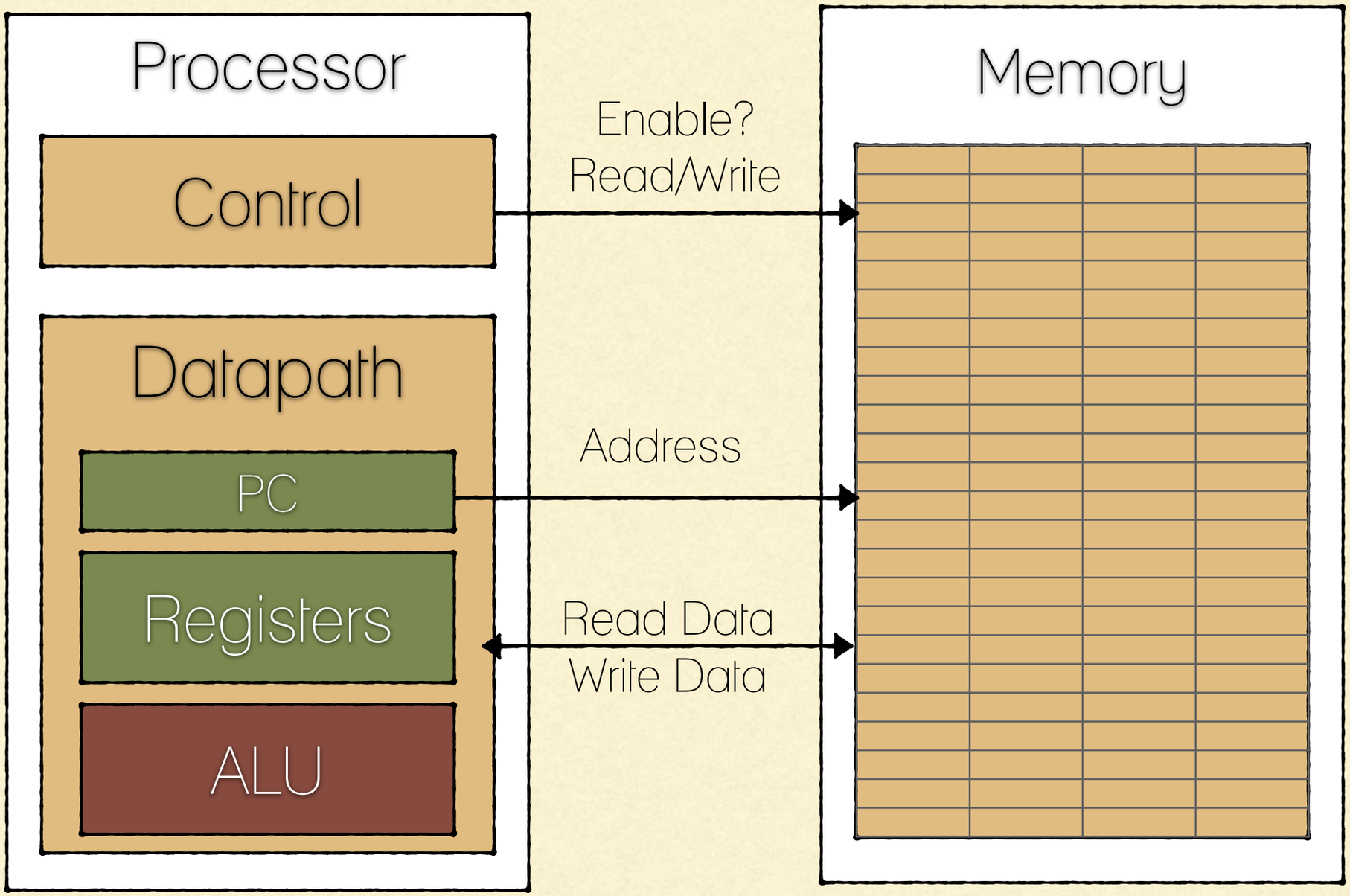
MIPS ASSEMBLY PROGRAMMING LANGUAGE PART II

Ayman Hajja, PhD

MEMORY ADDRESSES ARE IN BYTES



REGISTERS

- Unlike C or Java, assembly does not use variables
- Assembly operands are registers:
  - Limited number of special locations built directly into the hardware
  - Operations can only be performed on registers
- Since registers are are directly built in the CPU, they are very fast (100 to 500 times faster than main memory)

REGISTERS

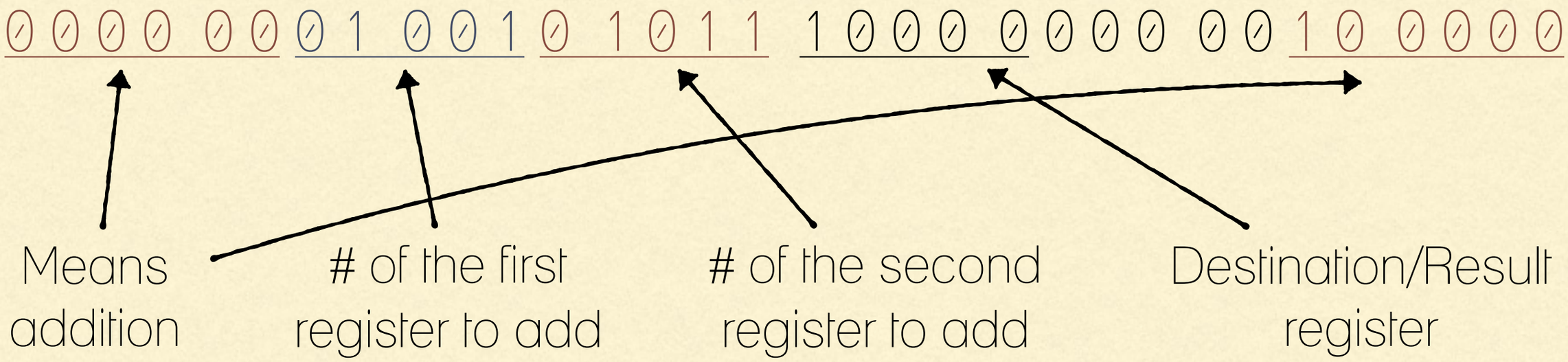
- Since registers are directly built in the CPU, there is a predetermined number of them:
  - In MIPS, we have 32 general-purpose registers, and few other special-purpose registers

REGISTERS

- Registers are numbered from 0 to 31
- Each register can be referred to by a number or name
- Number references:
  - $0, $1, $2, …, $30, $31
- For now:
  - $16 to $23 will be referred to by $s0 to $s7 (variables)
  - $8 to $15 will be referred to by $t0 to $t7 (temp variables)
- In general, use names to make your code more readable

MACHINE INSTRUCTIONS

- A machine instruction is a pattern of bits that directs the processor to perform one machine operation.
- Here is the machine instruction that directs the MIPS processor to add two 32-bit registers together (a register is a part of the processor that holds a bit pattern).



MIPS INSTRUCTIONS: ADD (REGISTER INSTRUCTION)

- Addition in Assembly:
  - Example 1: add $s0, $s1, $s2 (in MIPS)
    - Equivalent to a = b + c; (in C), assuming that:
      - $s1 contains the value of b
      - $s2 contains the value of c
      - and $s0 will be used to store the result (equivalent to 'a')

MIPS INSTRUCTIONS: ADD (REGISTER INSTRUCTION)

- Addition in Assembly:
  - Example 1: add $s0, $s1, $s2 (in MIPS)
  - Equivalent to a = b + c; (in C), assuming that:
    - $s1 contains the value of b
    - $s2 contains the value of c
    - and $s0 will be used to store the result (equivalent to 'a')
  - Example 2: add $s0, $s1, $zero (in MIPS)
  - Equivalent to f = g; (in C), assuming that $s0 corresponds to f, and $s1 corresponds to g

MIPS INSTRUCTIONS: SUB (REGISTER INSTRUCTION)

- Subtraction in Assembly:
  - Example: sub $s3, $s4, $s5 (in MIPS)
  - Equivalent to: d = e - f; (in C), assuming that:
    - d corresponds to $s3
    - e corresponds to $s4
    - f corresponds to $s5

MIPS INSTRUCTIONS: ADDITION AND SUBTRACTION OF INTEGERS

- How to do the following C statement?

  a = b + c + d - e;

a	$s0
b	$s1
c	$s2
d	$s3
e	$s4

MIPS INSTRUCTIONS: ADDITION AND SUBTRACTION OF INTEGERS

- How to do the following C statement?

  a = b + c + d - e;

- We break it into multiple instructions:

```
add $t0, $s1, $s2    # temp = b + c
```

a	$s0
b	$s1
c	$s2
d	$s3
e	$s4

MIPS INSTRUCTIONS: ADDITION AND SUBTRACTION OF INTEGERS

- How to do the following C statement?

  a = b + c + d - e;

- We break it into multiple instructions:

```
add $t0, $s1, $s2    # temp = b + c
add $t0, $t0, $s3    # temp = temp + d
```

a	$s0
b	$s1
c	$s2
d	$s3
e	$s4

MIPS INSTRUCTIONS: ADDITION AND SUBTRACTION OF INTEGERS

- How to do the following C statement?

```
a = b + c + d - e;
```

- We break it into multiple instructions:

```
add $t0, $s1, $s2    # temp = b + c
add $t0, $t0, $s3    # temp = temp + d
sub $s0, $t0, $s4    # a = temp - e
```

a	$s0
b	$s1
c	$s2
d	$s3
e	$s4

IMMEDIATES

- Immediates are numerical constants that are embedded in the instruction itself
- Add Immediate:

addi $s0, $s1, -10 (in MIPS)

f = g - 10; (in C)

assuming $s0 and $s1 are associated with the variables f, g respectively