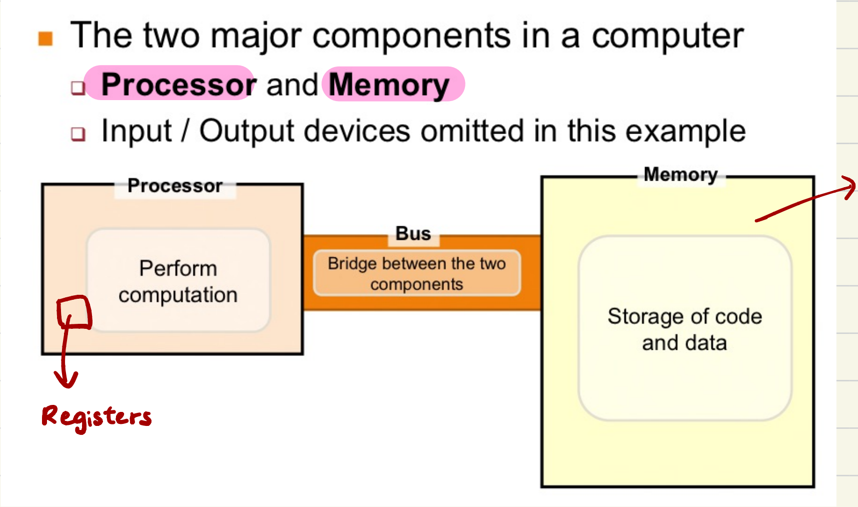
CS2100 - L07 - MIPS: Introduction Week 3+4

- Execution Walkthrough

- Simple MIPS Instructions
  - Arithmetic Operations
  - Immediate Operands
  - Logical Operations

Execution Walkthrough

Memory access : slow (~50ns)
Registers : fast (1ns / instruction)



- The two major components in a computer
  - **Processor** and **Memory**
  - Input / Output devices omitted in this example

Processor

Perform computation

Bus

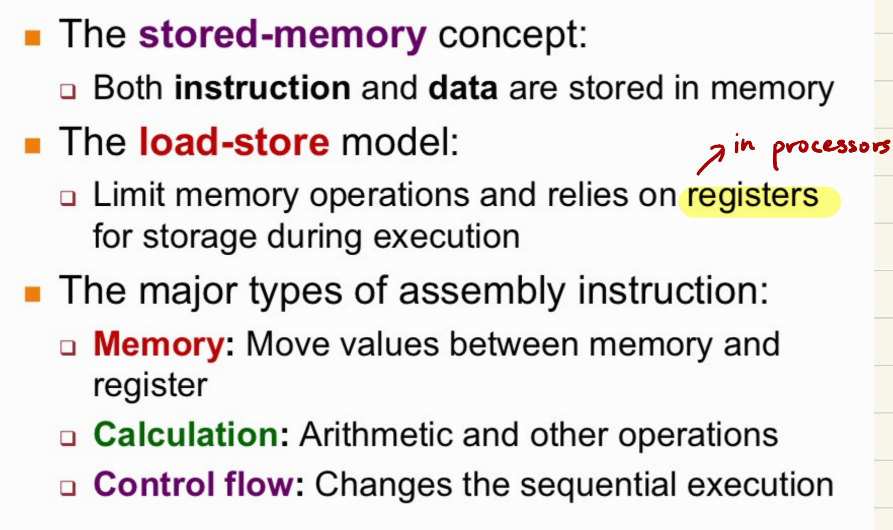
Bridge between the two components

Memory

Storage of code and data

Registers

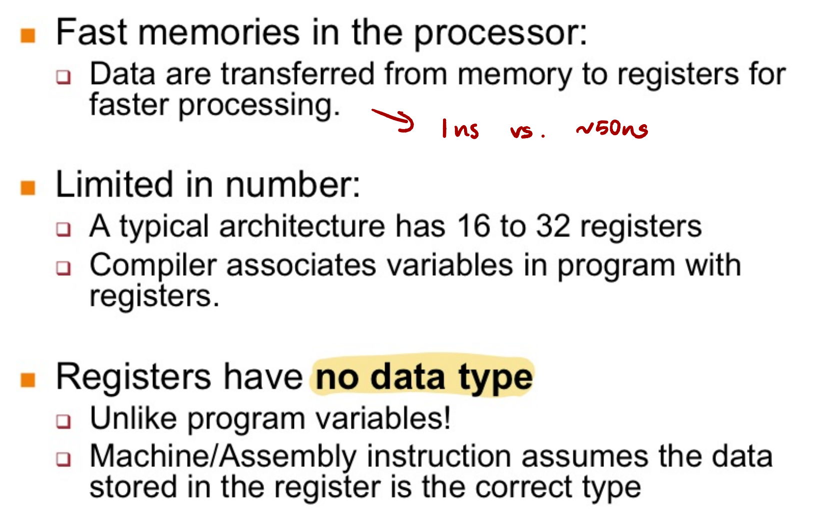
Long single 1D array of bytes/words



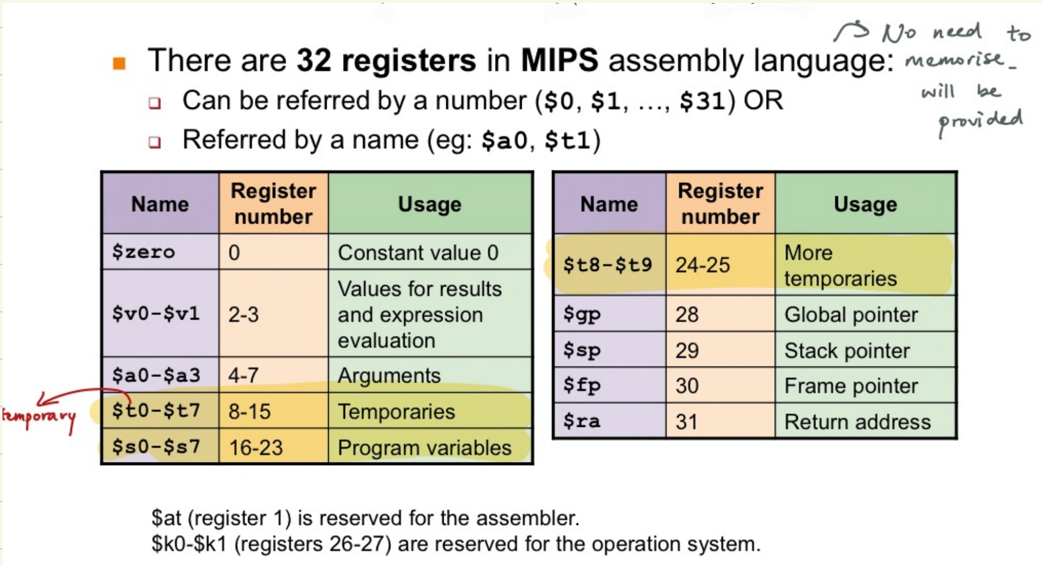
- The **stored-memory** concept:
  - Both **instruction** and **data** are stored in memory
- The **load-store** model:
  - Limit memory operations and relies on registers for storage during execution
- The major types of assembly instruction:
  - **Memory**: Move values between memory and register
  - **Calculation**: Arithmetic and other operations
  - **Control flow**: Changes the sequential execution

in processors

General Purpose Registers



- Fast memories in the processor:
  - Data are transferred from memory to registers for faster processing. → 1ns vs. ~50ns
- Limited in number:
  - A typical architecture has 16 to 32 registers
  - Compiler associates variables in program with registers.
- Registers have **no data type**
  - Unlike program variables!
  - Machine/Assembly instruction assumes the data stored in the register is the correct type



- There are **32 registers** in **MIPS** assembly language:
  - Can be referred by a number (**`$0`**, **`$1`**, …, **`$31`**) OR
  - Referred by a name (eg: **`$a0`**, **`$t1`**)

↳ No need to memorise_ will be provided

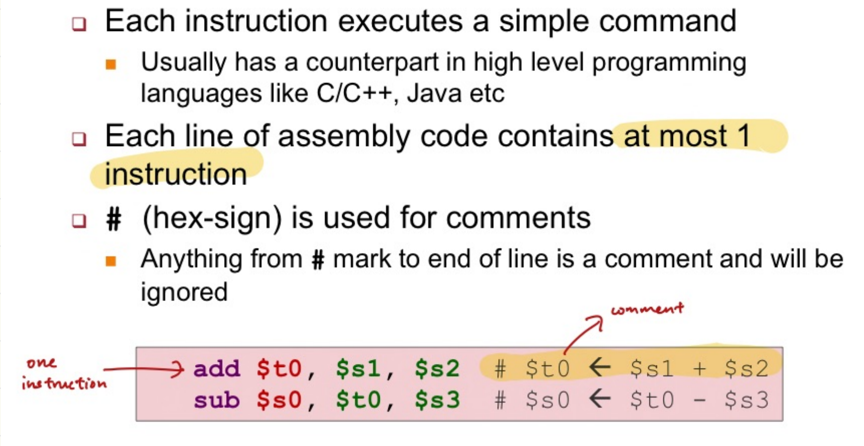
Name	Register number	Usage
`$zero`	0	Constant value 0
`$v0-$v1`	2-3	Values for results and expression evaluation
`$a0-$a3`	4-7	Arguments
`$t0-$t7`	8-15	Temporaries
`$s0-$s7`	16-23	Program variables

temporary

Name	Register number	Usage
`$t8-$t9`	24-25	More temporaries
`$gp`	28	Global pointer
`$sp`	29	Stack pointer
`$fp`	30	Frame pointer
`$ra`	31	Return address

$at (register 1) is reserved for the assembler.
$k0-$k1 (registers 26-27) are reserved for the operation system.

MIPS - Assembly Language



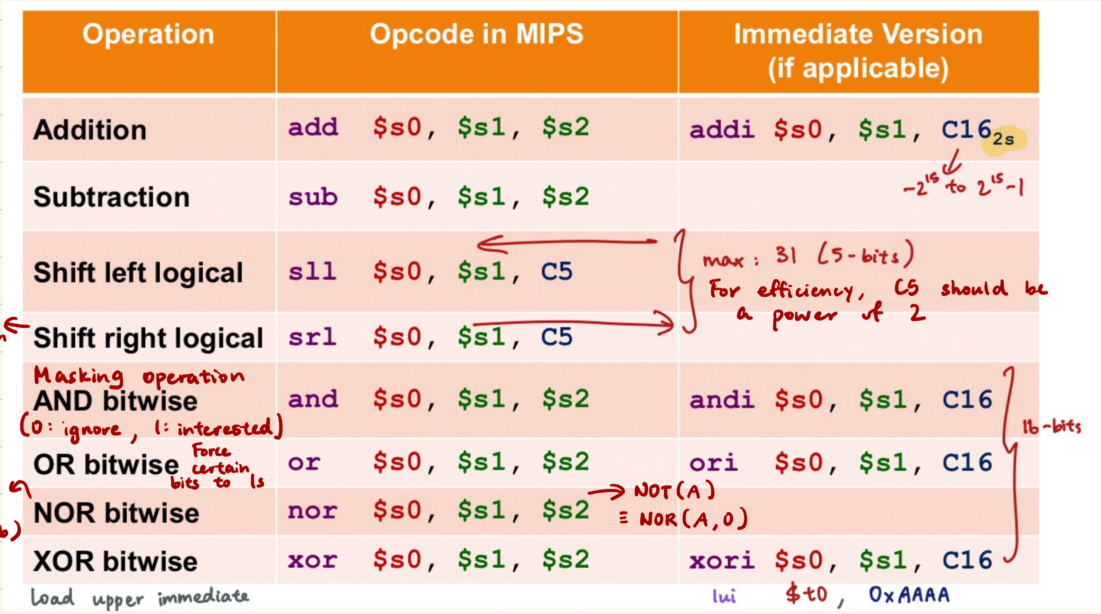
- Each instruction executes a simple command
  - Usually has a counterpart in high level programming languages like C/C++, Java etc
- Each line of assembly code contains at most 1 instruction
- # (hex-sign) is used for comments
  - Anything from # mark to end of line is a comment and will be ignored

comment

one instruction

```
add $t0, $s1, $s2   # $t0 ← $s1 + $s2
sub $s0, $t0, $s3   # $s0 ← $t0 - $s3
```

MIPS - Basic Instructions



Operation	Opcode in MIPS	Immediate Version (if applicable)
Addition	add $s0, $s1, $s2	addi $s0, $s1, C16 2s
Subtraction	sub $s0, $s1, $s2	
Shift left logical	sll $s0, $s1, C5	
Shift right logical	srl $s0, $s1, C5	
AND bitwise	and $s0, $s1, $s2	andi $s0, $s1, C16
OR bitwise	or $s0, $s1, $s2	ori $s0, $s1, C16
NOR bitwise	nor $s0, $s1, $s2	
XOR bitwise	xor $s0, $s1, $s2	xori $s0, $s1, C16

-2^{15} to $2^{15}-1$

max : 31 (5-bits)

For efficiency, C5 should be a power of 2

int division

Masking operation

(0: ignore, 1: interested)

Force certain bits to 1s

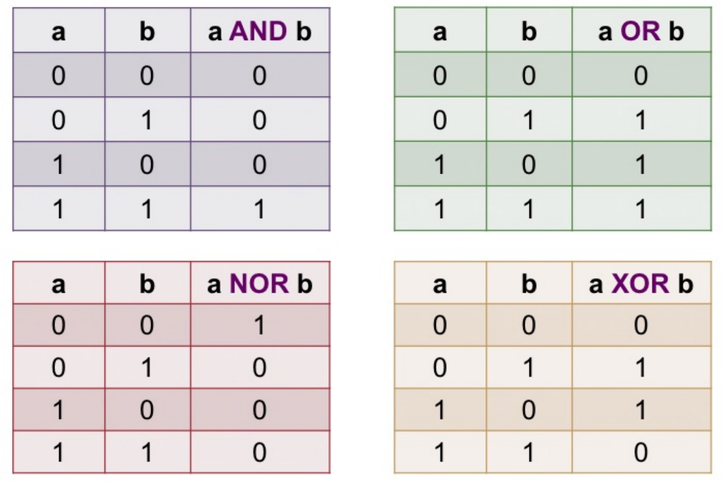
NOT (a+b)

→ NOT(A) ≡ NOR(A,0)

16-bits

Load upper immediate lui $t0, 0xAAAA

Logical Operations - Truth Tables



a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

a	b	a NOR b
0	0	1
0	1	0
1	0	0
1	1	0

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0