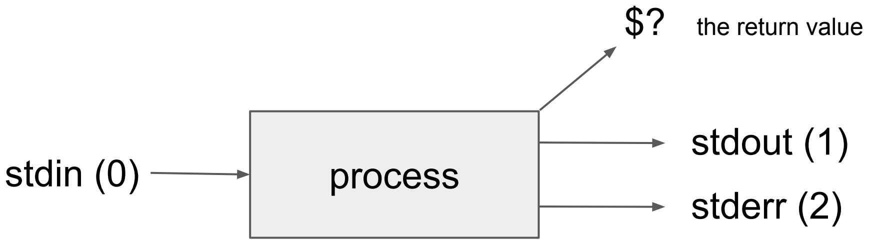
The Process and Standard File Descriptors (fds)



$? the return value

stdin (0)

process

stdout (1)

stderr (2)

Java Parlance:	System.in	== stdin
	System.out	== stdout
	System.err	== stderr

Memory Organization (java program)

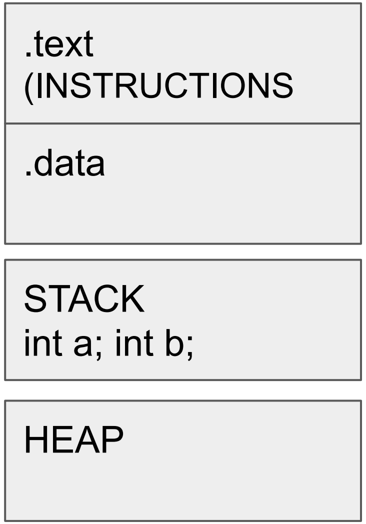
```
class Main {

 public static int x = 5;
 int y = 7;

 public int addNumbers(int a, int b) {
    int sum = a + b;
    return sum;
  }

  public static void main(String[] args) {
       int num1 = 25;
       int num2 = 15;

   // create an object of Main
    Main obj = new Main();
    int result = obj.addNumbers(num1, num2);
    System.out.println("Sum is: " + result);
  }
}
```



.text
(INSTRUCTIONS

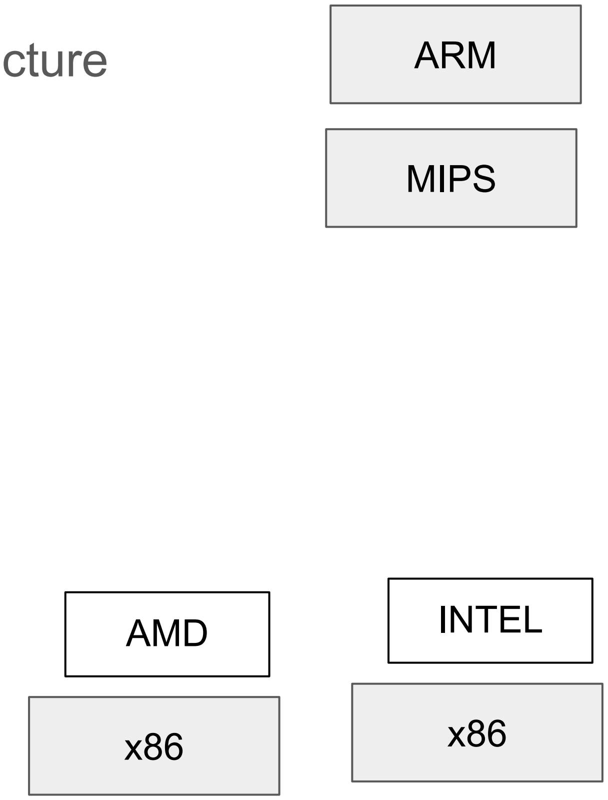
.data

STACK
int a; int b;

HEAP

Instruction Set Architectures

- The ISA is one level above the physical architecture
- Defines the following:
  - Supported instruction and their semantics
  - Supported data types
  - Registers: size, number, and purpose
  - Memory: layout, addressing, alignment, endiance
  - OS interface:
- Does not define the following:
  - technology used
  - chip layout
  - memory implementation
  - etc
- Very Similar to an API:
- RISC versus CISC



ARM

MIPS

AMD

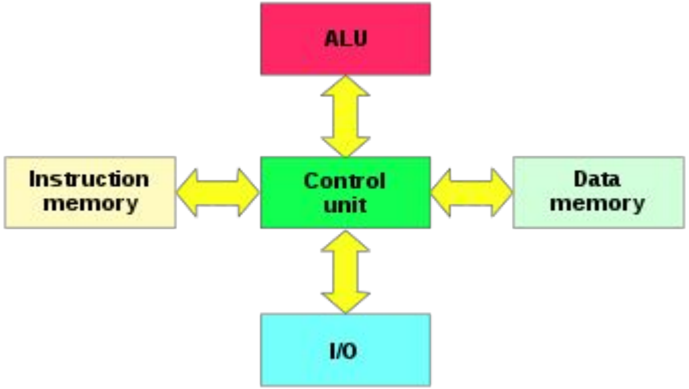
INTEL

x86

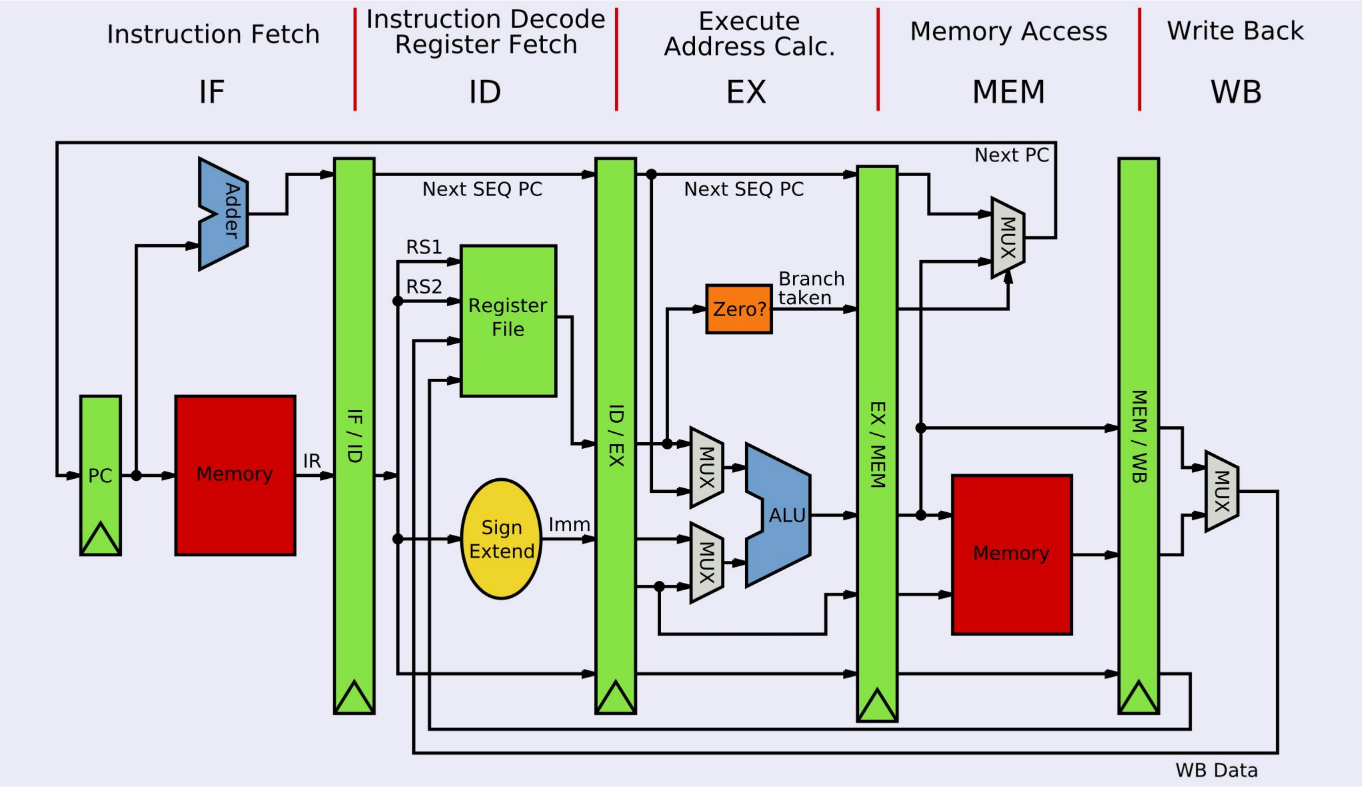
x86

Generalized Execution Cycle

1. Fetch:
   - Move the instruction into the control unit
2. Decode
   - Set control lines to allow data to flow to ALU
3. Execute
   - Activate the ALU
4. Writeback
   - Write the data to a register or memory



MIPS Microarchitecture



MIPS Pipeline Execution

Instruction	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8
#1	Fetch	Decode	Execute	Mem	WB			
#2		Fetch	Decode	Execute	Mem	WB		
#3			Fetch	Decode	Execute	Mem	WB	
#4				Fetch	Decode	Execute	Mem	WB
#5					Fetch	Decode	Execute	Mem

MIPS ISA Architecture: Instructions

- Three basic instruction types
  - Arithmetic, bitwise logic, etc.
  - Data transfers
  - Basic control flow

- Examples:
  - add $v0, $v0, $a0 ; $v0 = $v0 + $a0
  - sll $a0, $a1, $a2 ; $a0 = $a1 << $a2

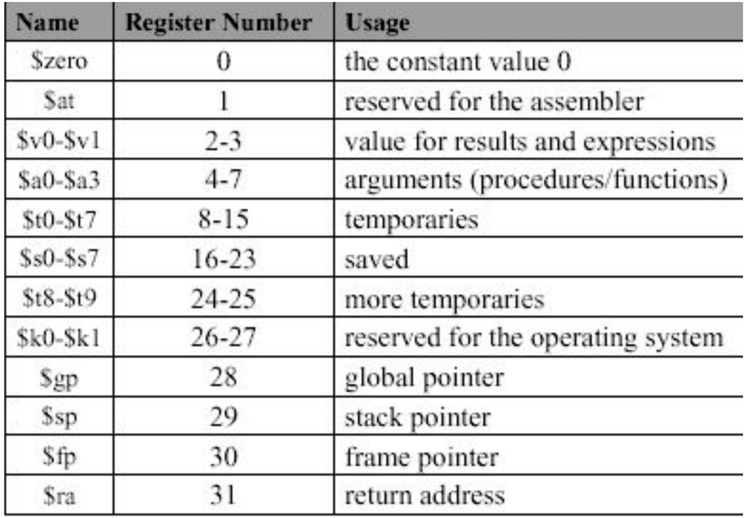
  - mov $t1, $t2 ; $t1 = $t2
  - lw $s0, 0($v0) ; $s0 = MEM[$v0 + 0 .. $v0 + 0 + 3] ; $s = 0($v)
  - lh $s0, 0($v0) ; $s0 = MEM[$v0 + 0 .. $v0 + 0 + 1]

  - beq $t3, $t5, label ; if ($t3 == $t5) goto label
  - jal proc ; call proc or proc()

MIPS ISA Architecture: Registers

- Data types:
  - byte, half, word
  - integer (signed/unsigned), binary32, binary64
- Registers:
  - 32: 32-bit integer registers
  - 32: 32-bit floating point registers
    - binary32: $fp0 .. $fp31
    - binary64: {$fp0, $fp1} .. {$fp30, $fp31}
  - 3: system registers: pc, hi, lo

Integer Registers

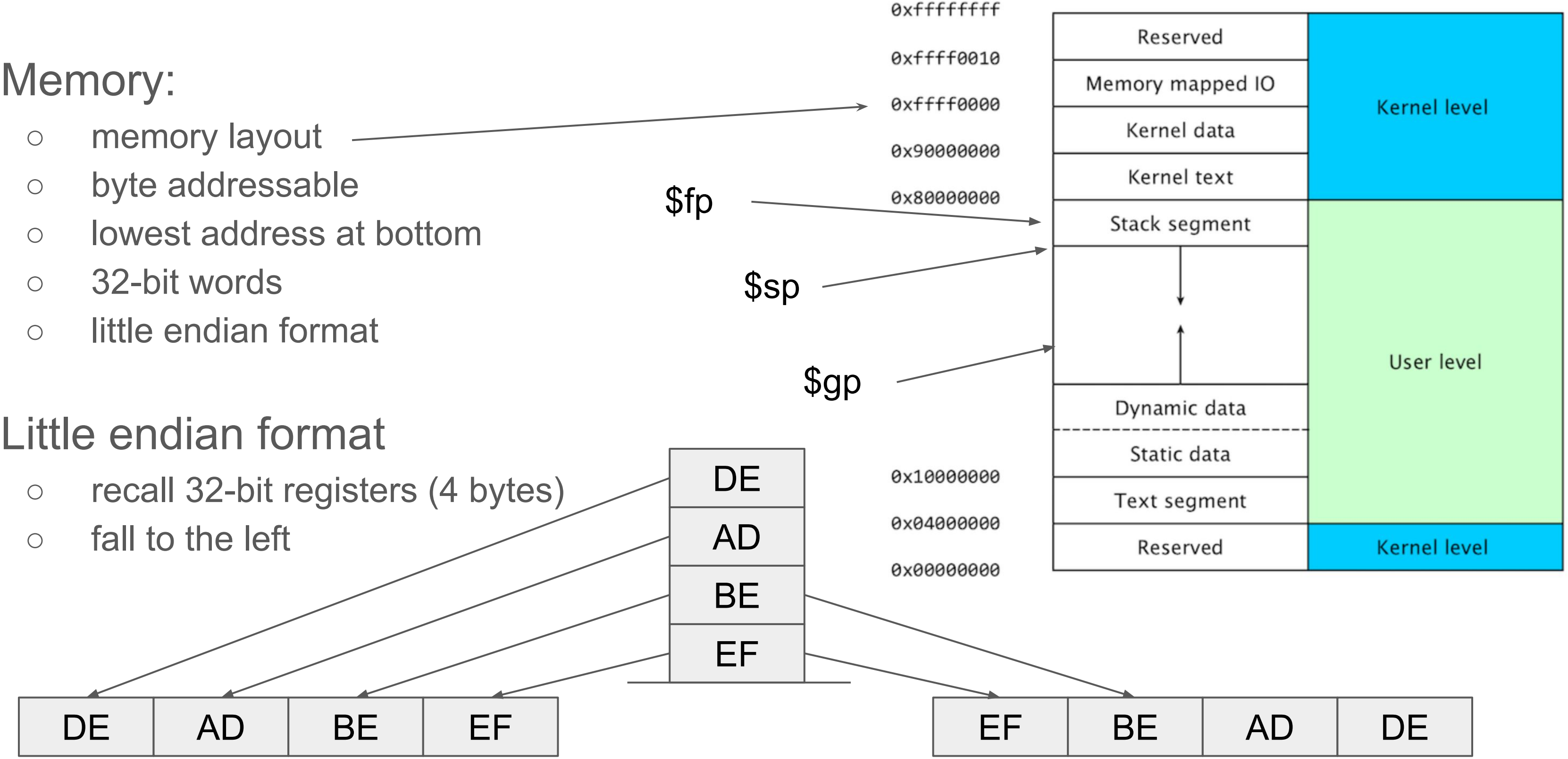


Name	Register Number	Usage
$zero	0	the constant value 0
$at	1	reserved for the assembler
$v0-$v1	2-3	value for results and expressions
$a0-$a3	4-7	arguments (procedures/functions)
$t0-$t7	8-15	temporaries
$s0-$s7	16-23	saved
$t8-$t9	24-25	more temporaries
$k0-$k1	26-27	reserved for the operating system
$gp	28	global pointer
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	return address

MIPS ISA Architecture> Memory Layout

- Memory:
  - memory layout
  - byte addressable
  - lowest address at bottom
  - 32-bit words
  - little endian format

- Little endian format
  - recall 32-bit registers (4 bytes)
  - fall to the left



MIPS ISA Architecture: OS interface

- System Calls: 'syscall' instruction

Service Name	$v0	input: $a0..$a3	output: $v0..$v1
print integer	1	$a0 = value	none
read integer	5	none	$v0 = value
malloc	9	$a0 = size	$v0 = buffer address
exit	10	none	none
file read	14	$a0 = fd, $a1 = buffer address $a2 = num bytes	$v0 = bytes read -1 == error 0 == eof