SysPrintInt	1
SysPrintFloat	2
SysPrintDouble	3
SysPrintString	4
SysReadInt	5
SysReadFloat	6
SysReadDouble	7
SysReadString	8
SysAlloc	9
SysExit	10
SysPrintChar	11
SysReadChar	12
SysOpenFile	13
SysReadFile	14
SysWriteFile	15
SysCloseFile	16
SysExitValue	17
SysPrintIntHex	34
SysPrintIntBin	35
SysRandInt	41
SysRandIntRange	42

CPU Time = CPU Clock Cycles / Clock Rate
= CPU cycles×Clock Period
= (IC * CPI) / Clock Rate(Hz)

Clock Period = 1 / Clock Frequency(Hz)

CPI = CPU Clock Cycles / IC
Average CPI = CPI(i) * IC*(i) / Total IC
Clock Cycles=Instruction Count (IC)×CPI

Performance is improved by
Reducing number of Clock Cycles (IC * CPI)
Increasing Clock Rate (Clock Frequency)
X is n time faster than Y"
PerformanceX / PerformanceY
= CPU timeY / CPU timeX = n

C Program -> Compiler -> Assembly language program -> Assembler -> Object: Machine / Object: Library -> Linker -> Executable: Machine -> Loader -> Memory

Compiler: high-level code to assembly code
Assembler: assembly code to machine code
Linker: Resolves labels in independent code and creates an object file
Loader: places an object program into main memory

Command Formats:
sw $src, offset($dst)
lw $dst, offset($src)
slt $t0, $t1, $t2 $t0 = ($t1 < $t2) ? 1 : 0
slti $t0, $t1, 5 $t0 = ($t1 < 5) ? 1 : 0
lb dest, offset(src)
sb src, offset(dest)

Leaf procedure: does not call another function.
Non-leaf: calls others → must save $ra, $s registers on stack.
Recursive: calls itself → must manage stack carefully.
For Procedures: $s# must be saved, $a#, $v#,$t# can be overwritten

2^0 = 1
2^1 = 2
2^2 = 4
2^3 = 8
2^4 = 16
2^5 = 32
2^6 = 64
2^7 = 128

Abstraction is simplifying systems by hiding details and focusing on relevant aspects

floating-point = IEE 754
fixed-point = fraction

Execute a software program, the microprocessor "fetches" each instruction from memory, "decodes" it, then "executes" it.

Clock Cycle (Clock Period) - for pipeline its longest time for one single step
Latency - Time for total instruction (no pipeline latency = clock period) and pipeline = clock cycle * num of steps
Throughput = 1 / clock period (in giga 10^9)

Time (sec)
10^-3 = milli (ms)
10^-6 = micro (µs)
10^-9 = nano (ns)
10^-12 = pico (ps)

Two's complement Fixed Point Addition is similar to Integer Addition
Before you add, line up the binary point
Zero-extend on the LSB side, Sign-extend on the MSB side

In single float point instruction its every 4 bytes and in double its every 8 so arr[3]
In single is 12($0)
In double is 24($0)

Unsigned - Always Postive
Signed - MSB is the positive or negative indicator (1 is negative 0 is positive)
Two's Complement - The MSB is always negative 1000 (-2^3)

Conditional flag bits 0 = false, 1 = true

c.lt.s $f1, $f2
bc1t L1 if (f1 < f2) branch

Answers for control logic bits should be given as 0 and 1

MIPS Multiplication:
32-bit x 32-bit produces a 64-bit product
HI: most-significant 32-bits (mfhi rd) HI to rd
LO: least-significant 32-bits (mflo rd) LO to rd
LO is from byte 0 to 7 and HI is bytes 8-15
MIPS: Division
div rs, rt / divu rs, rt. HI = remainder / LO = quotient

8 bits in a byte
Variables stored in .data
.asciiz for strings
.eqv sub second operand for first

R,I(lw),I(sw),I(beq),I(ALU) all use ALU & Register File & Instruction MEM

I(lw/sw) use Data Memory

32 bit hex is 0x0000_0000

A << N = A * 2^n (left / logical)
A >>> N = A / 2^n (this arithmetic)

Each register is 4 bytes

Shift Summary:
Logical: Replace with 0's
Arithmetic: Keep sign bit