Chapter 1

Moores Law – intergrated circuits resource double every 18-24 months.

Abstraction – represent the design at different levels of representation; lower level details are hidden to offer a simpler model at higher levels

Parallelism – performing operations in parallel

Pipelining – pattern of parallelism that allows the code to flow through the pipe continuously. Allowing for each step to be constantly working on code

Prediction – to guess and start working on the next step, instead of waiting for a response

Hierarchy of memories – fastest. Smallest, and most expensive memory per bit at the top of the hierarchy and the slowest, largest and the cheapest bit of memory at the bottom

Dependable and redundant – ensures the device cannot fail by including redundant components that can take over when a failure occurs and help detect failures

Throughput/bandwidth - the number of tasks completed per unit time

Performance = ( 1 / execution time )

Per(a) / Per(b) = Exe(b) / Exe(a) = n IT’S A RATIO

CPU Execution time for a program = CPU clock cycles for a program x clock cycle time

Improving performance = CPU Time = ( CPU clock cycles / Clock rate)

Instruction Performance = CPU clock cycles = instructions for a program x Average clock cycles per instruction

Classic CPU performance equation = CPU time = ( instruction count x CPI ) / Clock rate

Amdahl's Law = A rule stating that the performance enhancement possible with a given improvement is limited by the amout that the improved feature is used. It is a quantative version of the law of diminishing returns.

Execution time after improvement = Exectution time affected by improvement / amount of improvement + exection time unaffected

Example: Execution time after improvement = 80 sec / n + (100-80sec)

=20 seconds = (80 sec/ n) + 20 seconds == 0 = 80sec / n Therefore no amount enhanced

MIPS – Million instructions per second = MIPS = (Instruction count / Execution time x 10^6)

MIPS = (instruction count / ((instruction count x CPI)/clock rate) x 1-\*6) = (clock rate/ (CPIx10^6)

(Seconds / Program) = (instructions/program) x (clock cycles / instruction) x (second / clock cycle)

Chapter 2

Mips Example code:

Add a, b, c # The sum of b and c is placed in a

Add a, a, d # the sum of b c and d is now in a

Add a, a, e # now the sum of b c d and e are in a

Translating c to mips

A = b + c; -----> add a, b, c d = a-e; -----> sub d, a, e

F = (g+h)-(I+j) ----> add t0, g, h add t1, I, j, sub f, t0, t1

G = h + A[8] -----> lw $t0, 8($s3) # temp reg $t0 gets a[8]. ----> add $s1, $s2, $t0 # h and a[8] sum is stored in g

|  |  |
| --- | --- |
| Add $t0, $s6, $s0 | T0 = a[f] |
| Add $t1, $s7, $s1 | T1 = B[g] |
| Lw $s0, 0($t0) | F = a[f] |
| Addi $t2m $t0, 4 | T2 = a[f] + 4 |
| Lw $t0, 0($t2) | T0 = a[f]+4 |
| Add $t0, $t0, $s0 | T0 = a[f]+4 + a[f] |
| Sw $t2, 0($t1) | T2 = b[g] |

Chapter 3

Over flow – the exponent is too larger to be represented in the exponent field

Underflow – occurs when the negative exponent is too large to fit in the exponent field

Double precision – represents in two 32 bit words.

Single precision – floating point value represented in a single 32 bit word

13.2e-30 x 3.5e25 is an example of underflow because to denormalize the 3.5e25 to the smaller number you would lose all the significant figures. Cannot be rewritten

3.5e25 x 3.5e25 / 12.20 is overflow and cannot be rewritten

13.2e20 / 3.5e25 + 5.7e8 is overflow because to normalize the numbers the bigger numbers grow too big to be represented. Can be rewritten

40 bit wordsize – RISC – MIPS – 200 instruction codes. You have 40 registers because simplicity favors regulority

Max number of instruction codes for machine is 2 raised to the number of commands it can issue before storing.