Performance Architecture

exercises

P. Bakowski

bako@ieee.org

Ex1: Basic performance

- The **execution time** for a given task is a reliable method to evaluate computer performance.

The execution time is given by the following equation:

ex_time = instructions/task*

clock_cycles/instruction*

seconds/clock_cycle

Ex1: Basic performance

1. **clock_cycle/instruction** must be considered as an **average value** (**explain why ?**)

2. Let us calculate **ex_time** for:

   instructions/program=**100 Gi**

   clock_cycles/instruction = **1.2**

   clock frequency = **2.4 GHz**

M-Ex1: Basic performance

instructions/program=100 Gi,

clock_cycles/instruction = 1.2 and

clock frequency = 2.4 GHz

$$ex\_time = 100 * 10^{9} * 1.2 * 1/2.4 * 10^{-9} =$$

$$100 * 1.2 / 2.4 = 50 \text{ s}$$

Ex2: Performance in MIPS

- Let us consider **n** instruction classes [max 4] and the corresponding **CPI** measurements (Clock Cycles per Instruction) [from 1 to 4 CPI] for two computers.
- Each measurement is relative to the same program.
- Assume that the computer clock rate is CR = 1 GHz.
- Which code sequence will execute faster according to MIPS and according to execution time ?

Ex2: Performance in MIPS

Number of CPI classes per computer, in each case we give the number of Gi in the class with CR = 1 GHz:

```
NGiA=[1,2,5,4]

NGiB=[3,6,1,3]
```

For each class the **CPI** is given as:

```
CPI =[1,2,3,4]
```

- What is the execution time for each program (A,B) ?
- What is the MIPS measure for A and B ?

M-Ex2: Performance in MIPS

NGiA =[1,2,5,4], NGiB =[3,6,1,3]

CPI =[1,2,3,4] ,
CR = 1 GHz

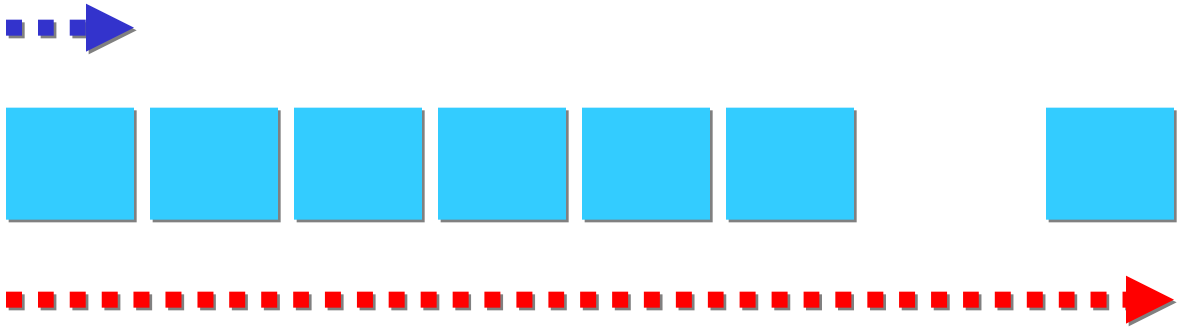
extA = 1*1*/CR + 2*2/CR + 5*3/CR + 4*4/CR = 36/CR

extB = 3*1*/CR + 6*2/CR + 1*3/CR + 3*4/CR = 30/CR

- What is the execution time for each program (A,B) ?
- What is the MIPS measure for A and B ?

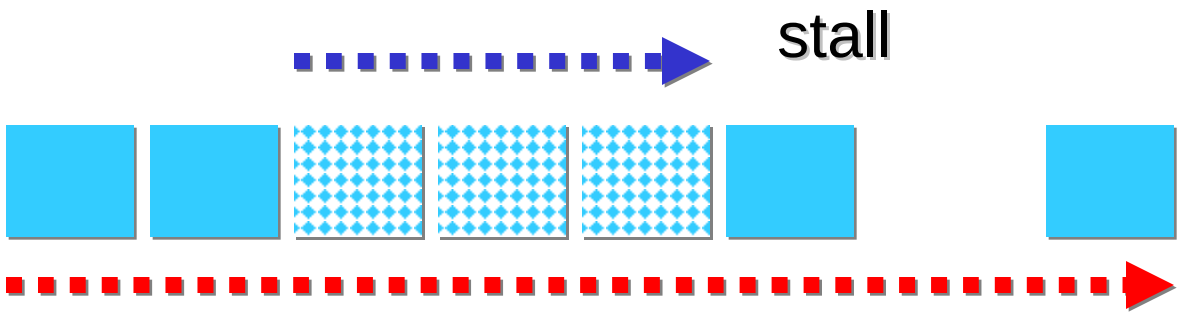
Ex3: Pipeline performance

- In this exercise we evaluate the execution speed up for a given pipeline.
- Each pipeline stage correspond to one clock cycle.
- The length of the pipeline is a parameter.



Ex3: Pipeline performance

- The executed program is a mix of instructions:
  - calculation instructions may be executed without pipeline stall
  - conditional branch instructions involve bps stages of delay (stall)
  - memory access instruction involve mps stages of delay



Ex3: Pipeline performance

- Given the clock frequency, the number of stages, the number of instructions to execute and the proportion of conditional/memory_access/calculation instructions,

  we can evaluate the necessary execution time for non-pipelined and pipelined architectures.

Ex3: Pipeline performance

parameters

- clock frequency in MHz: 1000
- number of pipeline stages: 8
- total instruction number in millions (inb): 1000
- ratio of branch instructions : 0.2
- average stall penalty for branch instructions : 6
- ratio of memory access instructions : 0.2
- average stall penalty for memory access instructions : 4

Ex3: Pipeline performance

Non pipelined execution time is given as:

npipe_time = clock_cycle * number_of_stages * number_of_instructions

Pipelined execution time is calculated as:

calrt = 1.0 - brrt – memrt

caltr - ratio of calculation instructions

pipe_time = clock_cycle *inb*calrt + clock_cycle *inb*brstall*brrt + clock_cycle *inb*memstall*memrt

M-Ex3: Pipeline performance

Non pipelined execution time is given as:

$$npipe\_time = 10^{-9} * 8 * 10^{9} = 8 \text{ s}$$

Pipelined execution time is calculated as:

$$calrt = 1.0 - brrt - memrt = 0.6$$

caltr - ratio of calculation instructions

$$pipe\_time = 10^{-9} * 10^{9} * 0.6 + 10^{-9} * 10^{9} * 6 * 0.2 + 10^{-9} * 10^{9} * 4 * 0.2 = 0.6 + 1.2 + 0.8 = 2.6 \text{ s}$$

Ex4: Power consumption

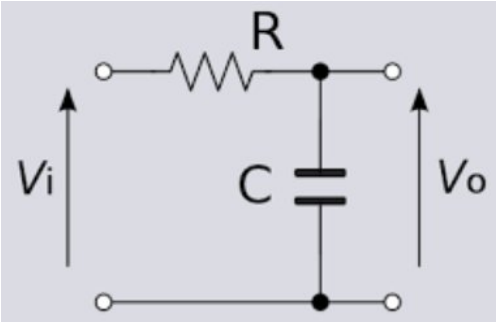
As we have seen in the performance equation, the easiest way to augment the performance is to increase the frequency.

The problem is here:

$$\textbf{dynamic.power} = A*N*C*V^2*f$$

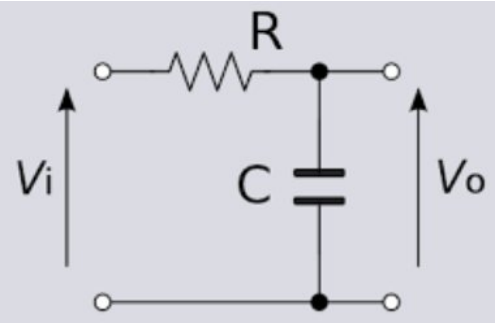
Apparently the dynamic power consumption (CMOS) is proportional to the frequency ?

But look at this ?



Ex4: Power consumption

dynamic.power = A*N*C*V^2*f



For the given circuit, if we wish to get the same V_o twice as quickly we must increment V_i by the factor **x=?**

$$V_o = x * V_i * (1 - e^{-t/\tau})$$

Let us fix the ratio V_o/V_i to 0.8, and τ to 0.1ns. What is the maximum switching frequency ?

Now let us try, for the same circuit, to double the switching frequency, what is the **x** factor.

M-Ex4: Power consumption

Let us fix the ratio V_o/V_i to 0.8, and τ to 1ns. What is the maximum switching frequency ?

$$V_o = x * V_i * (1 - e^{-t/\tau}) \Rightarrow$$

$$0.8 = (1 - e^{-t/0.1}) \Rightarrow 0.2 = e^{-t/1}$$

$$\Rightarrow \ln 0.2 = -t/1 \Rightarrow$$

$$t = -1 * \ln 0.2 = 1.61 \text{ ns}$$

frequency is 1/1.61ns = 0.62 GHz

M-Ex4: Power consumption

Now we wish to increase the frequency by the factor 2. So we need to reduce the t by the same proportion, it is
t=1.61 ns => 0.8 ns

$$V_o = x * V_i * (1 - e^{-t/\tau}) \Rightarrow$$
$$x = V_o / V_i * (1 - e^{-t/\tau}) = 0.8/(1 - e^{-t/\tau}) =$$
$$0.8/(1 - e^{-0.8}) = 0.8/(1-0.45) =$$
$$= 0.8/0.55 = 1.45$$

The resulting voltage increase x is: **1.45**

M-Ex4: Power consumption

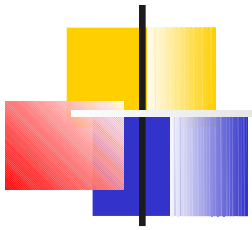
Coming back to our initial equation:

$$dynamic.power = A*N*C*V^2*f$$

Now we know that the increase of frequency by factor 2 involves the increase of the voltage by the factor – 1.45

So the total power increase (for the same circuit) is:

1.45*1.45*2 = 2.12*2= 4.24 times



Ex5: Amdahl's law

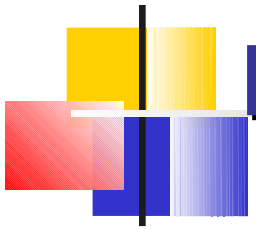
The execution time of a mixed, sequential-parallel program, after the acceleration of the parallel part, is given by a simple equation known as Amdahl's law.

tex_new = tex_par/acceleration + tex_seq

Calculate the necessary acceleration to obtain the required new execution time.

Take:

tex_par = 10 s, tex_seq = 5, tex_new = 8 s



Ex5: Amdahl's law

tex_new = tex_par/acceleration + tex_seq

Calculate the necessary acceleration to obtain the required new execution time.

tex_par = 10 s, tex_seq = 5, tex_new = 8 s

tex_par/acceleration = tex_new – tex_seq

acceleration = tex_par/ tex_new – tex_seq

acceleration = 10/(8 – 5) = 10/3 = 3.3

Summary

- Basic performance
- MIPS as performance measure
- Pipeline performance
- Power consumption
- Amdahl's Law