Dr: waleed Dueik

معمارية وتنظيم حاسوب
Computer Architecture and organization 1

12-10-2020

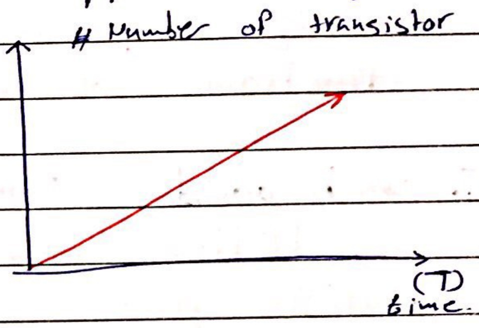
chapter 1 : computer Abstractions & Technology

* progress in computer technology supported by

* Moore's Law *

* Design for Moore's Law :

→ Transistor : اصغر شي موجود على ال perchip



More transistor → More gates → More (cpu)

every 18 months numbers of transistors Double.

↳ من ناحية الكل سنتين عدد ال Trasistors يتضاعف على نفس المساحة او بتقدر تحطيه نفس عدد ال Transistors بس بتقل المساحة.

* classes of computers

[1] personal computer

↳ general purpose

بتقدر تستخدمو لكل شي

↳ subject to cost / performance tradeoff

[2] server computers

↳ Network based

↳ High capacity , performance , reliability

(عدد الاخطاء اقل من ال (pc

↳ Range from small size to big size

[3] super computer ← تعتبر جديد

• ↳ "تعمل عمليات عالية جدا"

High-end scientific and engineering calculations

↳ تعم الرسائل العلمية من القوة

تحتاج الأجهزة القوية

↳ High capacity but represent small fraction of overall computer market

[4] embedded computers :- → أجهزة مدمجة

↳ Hidden components كأجزاء من أجهزة أخرى.

↳ High performance . cost constraints

(أداء عالي والتكلفة قليلة).

* The post pc era

[1] personal Mobile Device

- Battery operated
- connects to internet ex:- smart phones
- Hundred of Dollars - tablets

[2] cloud computing

- software as service
- ware house Scale computers (WSC)

↳ جزء من البرامج يشتغل على الـ cloud

والجزء الآخر على الـ ~~computer~~ computer

Ex:- Amazone and google.

Scanned with CamScanner

14-10-2020

Understanding performance :-

1) Algorithm → numbers of operations executed
   ↳ [illegible]
2) programming Language, compiler, architecture
   ↳ Determine number of Machine instructions executed per operations
3) processor & memory system
   ↳ Determine how fast instructions are executed
4) I/O system (including OS)
   (input/output) system

section 1.2 ⇒ ✱ Use abstraction to simplify Design.

↳ Represent the Design at Multiple Level such that low-Level Details are hidden
↳ increase productivity for computer architects and programs

case

✱ Make the common ↑ fast

cpu → sum ← 2 ns
cpu → Divid ← 20 ns

cpu 1
add → 1ns
divid → 18ns

cpu 2
add → 2ns
divide → 10 ns

cpu → Time = 90 x 2 + 10 x 26 = 386 ns
cpu1 → Time = 90 x 1 + 10 x 18 = 270 ns
cpu2 → Time = 90 x 2 + 10 x 1 = 280 ns

[*] performance via parallelism

← تحسين الاداء بتنفيذ المهام عن طريق تقسيمها الى اجزاء وجعلين run in parallel

← على التوازي بقلل الزمن ← اذا خرب جزء مابخرب باقي الاجزاء.

[*] performance via pipelining

← التنفيذ على مراحل

← كل ماتخلص مرحلة وتروح على المرحلة الي بعدها بنقدر نرجع نستخدم المرحلة الي قبل

مثال عليها :- الجسر المتحرك في عملية اطفاء الحريق

* performance via prediction

← نستخدم التنبؤ

IF (condition)
↓
else
|

* كلما زادت نسبة التنبؤ كلما استفدت اكثر

* نسبة النجاح > %50 ناجحة

" < %50 خاطئة وعليها نوايا

* better ask for forgiveness than ask for permisson

[*] Hierarchy of memories

← سرعة الكمبيوتر محددة بسرعة الذاكرة

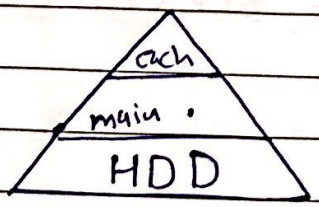
←

cpu ⇄ main memory ⇄ HDD

cpu ⇄ cash ⇄ main memory ⇄ HDD

Five Apple

ال cash ← صغيرة ، size صغير لذلك تكون سريعة.
* وبتحس انه كمان ال Data بناخدها من ال cash وال Main

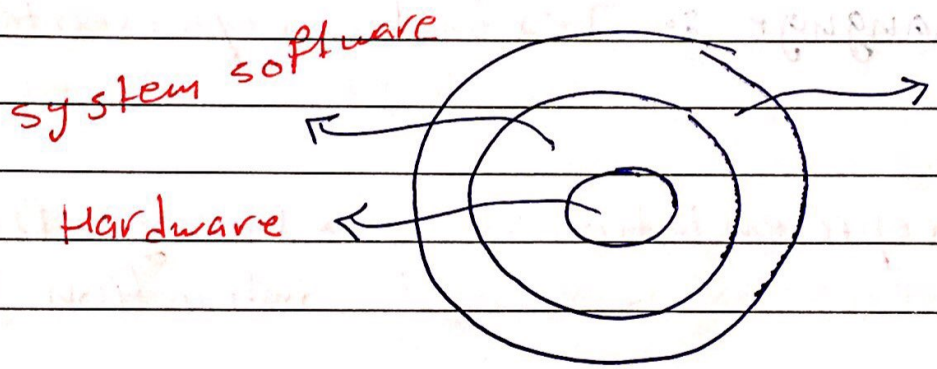


size ↑ speed ↓ cost ↓

* Depedability via redundancy.

← البيز از ← في حال فيه وحدة من ال coure ما بتاثر.

seelion 1.3



* Application software :: written in high-L
Languge (HLL)
↳ Improved productivly
↳ programs become independent from hardware.

* system soft ware
1) compiler → translate
2) operating system → service code

* Hardware
↳ processor , memory , I/O controllers

Levels of program code

[1] High - Level language :
provides for productivity & portability.

[2] Assembly language :- Textual representation of instruction

[3] Hard ware representation :-
* Binary Digits
* encoded instruction & Data

19-10-2020 Components of computers

* same components for all Kind of computer

* five component :- input / output / memory / data path and control

* Data path + control = cpu

* Input / output Devices
  - User - interface Devices: الأجهزة التي يستخدمها الـ user مثل keyboard أو Mouse
  - Storage Devices: Hard disk
    - Data: تستخدم لتخزين
  - Network adapters: للربط بين أجهزة الكمبيوتر

* one of very important Input/output Devices is LCD (Liquid crystal display) screen:
  - To Display images (pixels)
    - pixel ← كل صورة تتكون بداخله بكسلات
    - (pixels) الصورة الواحدة تتكون مجموعة من الـ
    - وتسمى بالـ a bit map

* size of Bit map is Based on: 1) screen size
  2) resolution
  - 1024 x 786 to 2048 x 1536

* Hardware support for graphics.
  - Frame buffer ← to store the bit map for the image to Display
    - الصورة اللي عايز عرضها على الشاشة تتكون
    - (Frame buffer) تتخزن في الـ

Touch screen:.

↳ postPc device

↳ ~~supersedes keyboard~~ supersedes keyboard & mouse

↳ Resistive & (capacitive) types

↳ allow multiple touches simultaneously.

* Inside The (cpu)

- Datapath
  ↳ performs operations on Data
  (Data) على البيانات في ال
- control
  ↳ sequences data path , memory
  ← وظيفتها: (التحكم)
- cach memory
  ↳ small fast SRAM for immediate access to Data

* عندما تكون ال cach داخل ال (cpu) تكون أسرع

A safe place for Data (1)

- Volatile main memory :: تفقد الـData عند انقطاع الجهاز
  → DRAMs : Dynamic Random access memory

• Non- volatile secondary memory ::

↳ Magnetic disk

↳ Flash memory in PMD's

↳ optical disk (CDROM, DVD)

ذاكرة حفظ البيانات Data

A safe place for Data (2)

↳ DRAM ↳ Magnetic Disk HDD ↳ Flash SSD

لا يجب ان نعرف ما هو الأسرع بينهم

↳

Past < slow < Moderate

* Network .. communication ربط الكمبيوتر بالانترنت

* Defining performance.

سرعة الركاب → Full كم يقطع → speed → سرعة الركاب بـ speed الـ

Response Time and Throughput

↳ Response or ~~exit~~ execution

الزمن الذي استغرقه لتنفيذ مهمة

↳ Throughput ← عدد الوظائف التي عملتها في زمن معين

Relative performance:

* Define performance = $\frac{1}{\text{execution time}}$

"X is n time faster than Y"

↳ performance x / performance y
= execution time y / execution x = n

example:- time taken to run a program

A = 10s

B = 15s

sol:- $\frac{\text{execution Time}_B}{\text{"}\quad\text{"}_A} = \frac{15\ s}{10\ s} = 1.5$

∴ A is 1.5 times faster than B

↳ $\frac{\text{performance A}}{\text{performance B}} = \frac{\frac{1}{15}}{\frac{1}{15}} = \frac{15}{5} = \boxed{3}$

21-10-2020 Measuring execution time.

* elapsed time ← هذا الوقت يشمل كل شيء.

↳ (الوقت الذي يقضيه البرنامج داخل ال cpu : operation I/O أو ال processing
ال idle time و ال)

↳ system performance = $\frac{1}{\text{elapsed time}}$

* cpu Time (10ns) "الزمن الذي يقضيه البرنامج وهو بداخل الـ cpu"

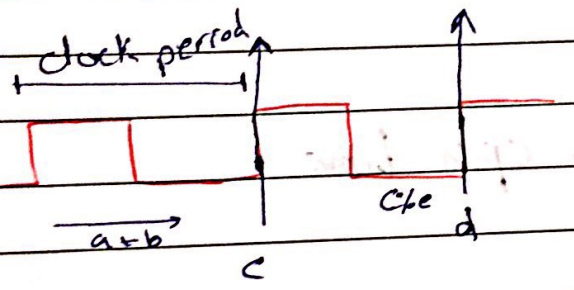
system cpu time (3 ns) ← OS الـ cpu الذي يقضيه

user cpu time (7ns)

cpu performance = $\frac{1}{\text{user cpu time}}$

* cpu clocking

↳ operation of digital hard ware governed by a constant-rate clock



c = a + b

D = c % e

syntax call x

* يستخدم الـ cycle لحساب افرق الزمني

clock cycle time = clock period = 250 ps = 250×10^{-12}

clock frequency (rate): cycle per second

$\frac{1}{F}$

cpu time = cpu clock cycles x clock cycle time

= cpu clock cycles / clock Rate

To improve your cpu ~~time~~ performance → يقلل time

وكان اقل ال time ← يقلل ال cpu clock او يقلل ال cycle

clock cycle ال time

* لنا قللنا عدد ال cycles لحالة يزيد زمن ال cycle الواحدة

* example of cpu Time:

↳ computer A = 2GH clock , 10s cpu time .

↳ Designing computer B

- Aim for 6s cpu time
- can Do faster clock , but causes 1.2 x clock cycle

A & B ↩

clock cycle B = [1.2 x 10]

↳ How fast must computer B clock be ?

↳

solution: cpu time$_B$ = cpu clock cycle / clock Rate

$10 = ? / 2 \times 10^9$

clock cycle$_A$ = 20×10^9

$$\text{clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6s} = \frac{24 \times 10^9}{6s} = 4\ \text{GHz}$$

[Instruction count & cpI]

- clock cycles = instruction count x cycles per instruction

- cpu time = instruction count x cpI x clock cycle time

$$= \frac{\text{instruction count} \times \text{cpI}}{\text{clock Rate}}$$

الأشياء اللي بتأثر على ال instruction count :

١) algorithm

٢) ISA

٣) compiler

* Average cycle per instruction
- Determine by cpu Hardware
- Different cpI have Different instruction
(Average cpI affected by instruction Mix)

example
A → cycle time = 250ps
cpI = 2.0
B → cycle time = 500ps
cpI = 1.2
} same ISA & compiler

which is faster & by How much?

cpu time A = instruction count × cpI A × cycle time A
1 × 2.0 × 250 = 1 × 500 ps
A is faster. ↔

cpu Time B = 1 × 1.2 × 500 = 1 × 600 ps

cpu time B / cpu time A = 1 × 600 / 1 × 500 = 1.2

26-10-2020

* If Different instruction classes tak Different numbers of cycles

$$\text{clock cycle} = \sum_{i=1}^{n} (\text{CPI} \times \text{instruction count}_i)$$

* weighted Average CPI

$$\text{CPI} = \frac{\text{clock cycle}}{\text{instruction count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \frac{\text{instruction count}_i}{\text{instruction count}}\right)$$

Relative Frequency

* Alternative compiled code sequences using instruction in classes A, B, C

classes	A	B	C
CPI Per class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

sequence 1: IC = 5

clock cycle = 2x1 + 2x1 + 2x3

= 10

Avg CPI = 10/5 = 2.0

sequence 2: IC = 6

clock cycle: 4x1 + 1x1 + 1x3

= 9

Avg CPI = 9/6 = 1.5

26-10-2020

* same ISA & same compile - same Ici & same Ic

* same Hardware - same T or same F & same CPIi

* same ISA & same compiler & same hardware
same T same F & same cpIi & same Ic
The same كل شي داخل المعالج هون بيز

example: two compiled sequence of the same program are given below

↓ gives the instructure of each type Ic

	A	B	C
seq.1	1	2	4
seq.2	2	P	2

↓ this table gives the CPI

	A	B	C
CPI	1	2	3

Given the two sequence running on the same computer, what is the number of instruction of type B in seq.2 that will make seq.1 two time faster than seq.2 ?

$\frac{\text{cpu time}_2}{\text{cpu time}_1} = 2$ ⇒ $\text{cpu time}_2 = Ic_2 \times \text{cpu}_{\text{avrg}_2} \times T$

clock cycle$_2$ × T

$(\Sigma Ic_i \times cpI_i) \times T$

$(2 \times 1 + P \times 2 + 2 \times 3) \times T$

$2 + 2P + 6$

$(8 + 2P) \times T$

$$\frac{\text{CPU Time}_2}{\text{CPU Time}_1} = 2 = \frac{(2 + Pr_2 \times T)}{1 + 1 \times T}$$

$$\text{CPU Time}_1 = IC_1 \times CPI_{avg_1} \times T$$
$$\text{clock cycle}_1 \times T$$
$$\left(\sum IC_i \times CPI_i\right) \times T$$
$$(1 \times 1 + 2 \times 2 + 4 \times 2)\, T$$

[performance summary]

$$\text{CPU Time} = \frac{\text{Instruction}}{\text{program}} \times \frac{\text{clock cycles}}{\text{instruction}} \times \frac{\text{second}}{\text{clock cycle}}$$

→ Depends: 1) program language 2) algorithm 3) compiler affects I & CPI

↳ ISA : affects IC, CPI & clock period

↳ Microarchitecture design : affects CPI

↳ Hardware implementation and technology : affects clock period (T_c)

* cost / performance is improving

* Hierarchical layer of abstraction (in both Hard+Soft ware)

* execution time : The Best performance measure

chapter 2 instructions : Language of the compute
ISA

↳ Instruction represent of the compute language.
every ~~computer~~ Language has → instruction set
vocabulary

↳ Diffrent computes have Diffrent instruction set
↳ All computer are constructed from hardware technology based on similar underlying principles

* RISC : Reduce Instruction set computer
(Fixed Length . Fast execution . simple Functionality)
32 ↔
64 ↔

* CISC : complex instruction set computer
(variable Length . slow execution . comple Functionality)

Famous commercial ISA
↳ MIPS ← Designed since the (1980)c (RISC)

↳ ARM ← Designed since (1985) (RISC)
more than 14 billion cheps with ARM processor
(* The Most popular instruction set in the world)

↳ inte x86. (CISC) ← which power both the pc and

The RisK-5 Instruction set

↳ open ISA
↳ Similiar ISAs have larg share of embedded & core market

* (RISC-5) has three design principles

IC Risc > IC cisc	
Fixed length	variable length
simple Functionaly	complex Functionality
Fast execution	slow execution.

Arithmetic operation

↳ each arithmetic instruction performs only one operation and must always have exactly three operands.

Two source & one destination

```
add  a, b, c   # a = b+c
     ↑  ↳ sources
destination
```

* All arithmetic operation have this form

[eg] we want to add four Numbes (b, c, d and e)

```
add  a, b, c
add  a, a, d   → Three instruction are needed
add  a, a, e
```

a = b + c + d + e.

* Design principle 1 : simplicity favors regularity
- Regularity makes implementation simpler
- simplicity enables higher performance at lower cost.

* C code :

$P = (g+h) - (i+j)$

↳ compiled RISC-V code.

```
add t0 , g , h      // temp = g + h
add t1 , i , j      // temp t1 = i + j
sub P , t0 , t1     // P = t0 - t1
```

* في الاخير يقدر يغير الترتيب أما بالعكس ✗

$P = (g+h) - i - j$ ⟵

```
add f , g , h
sub f , f , i
sub f , f , j
```

2.3 Registe operands

↳ Arithmetic instructions use register operands

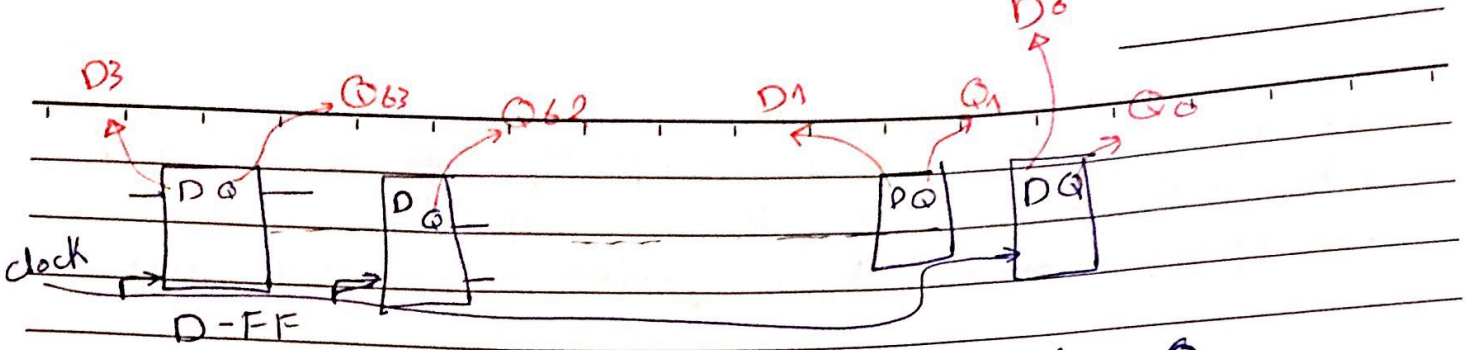
RISC-V has a 32 x 64 - bit register file.

↳ use for frequently accessed Data.

↳ 64-bit Data is called "Doubleword"

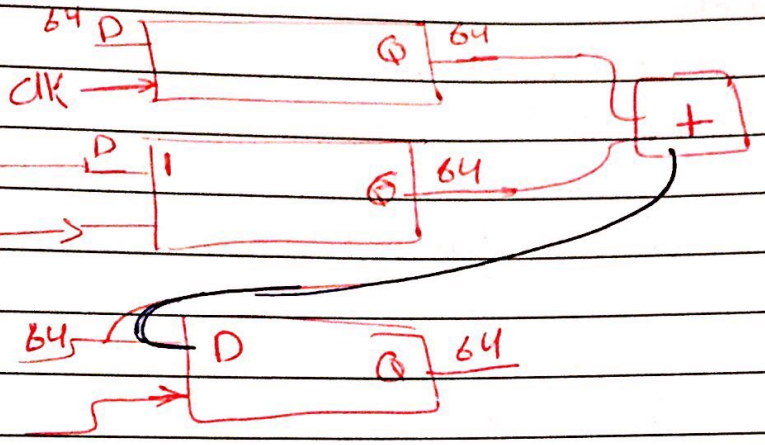
↳ 32 x 64 bit general pupose registers x0 to x31

* 32 bit Data called "word"



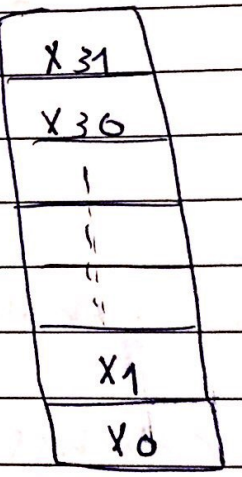
The value of D will be stored in Q

64-bit register



register file (RF)

SIZE = 32 x 64 bit



64-bit

8-bit = byte

32-bit = word

64-bit = Double word

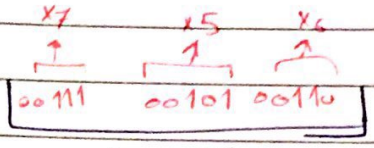
16bit = 1 half word

* Design principle 2: smaller is faster
↳ variable in HLL are unlimited while there are only 32 registers in RISC-V

RISC → instruction are 32-bit wide
↳ Data is 64-bit wide

add a,b,c

add x7, x5, x6	Machine code



32-bit

* Design must Balance the need for More registers By the programmes and the desire to keep the clock rate fast

* Numbes of registes also affect the Numbes of bit it would take to represent at speefic registe in the instruction format

[RISC-V Registes]

x_0	← The constant value.	* x12 - x17
x_1	← return address	
x_2	← stack pointer	
x_3	← global pointer	
x_4	← thread pointer	
~~x28-x31~~, x5-x7	← temporaries	
x_8	← fram pointe	
x9, x18 - x27	← functions arguments / results	
x10 - x11	← // //	

Five Apple