Plan

1. *Introduction, contexte*
2. Codage des nombres, des instructions
3. Exécution simple des instructions
4. Pipeline
5. Système mémoire
6. Caches
7. Entrées/Sorties

Références de ce cours

Livre

- Hennessy et Patterson, *Computer Architecture: a quantitative approach,* 5ème édition, Morgan Kauffman 2005

Cours en ligne

- CS 152, Berkeley, K. Asanovic
- CS 232 et CS 533, University of Illinois at Urbana Champaign, J. Torellas
- Architecture des ordinateurs, ENSEIRB, F. Pellegrini

Autre:

- ORAP, forum pour la promotion du parallélisme

Références de ce cours

- http://www.top500.org: articles et chiffres concernant les machines les plus puissantes
- http://www.itrs.net: International Technology Roadmap for Semiconductors
- http://www.arstechnica.com, http://www.realworldtech.com, http://www.wikipedia.org

Références de ce cours

Ces transparents contiennent des informations qui sont propriété de:

- Arvind (MIT)
- Krste Asanovic (MIT/UCB)
- Joel Emer (Intel/MIT)
- James Hoe (CMU)
- John Kubiatowicz (UCB)
- David Patterson (UCB)
- William Jalby (UVSQ)
- INTEL, ARM, IBM, Nvidia

2- Codage

Les ordinateurs ne manipulent que des 0 et 1.

a- les entiers naturels

b- les entiers relatifs

c- les réels

d- les instructions et programmes

e- implication sur ce que peut faire et ne pas faire un ordinateur

2- Codage

Les ordinateurs ne manipulent que des 0 et 1.

a- Les entiers

- Un bit (binary digit): vaut 0 ou 1
- N bits: 2^N valeurs possibles
- 8 bits: un octet (byte en anglais), 2^8 valeurs
- 1 Ko: un kilo octet = 1000 octets (KB en anglais)
- 1 Mo: 1000 Ko (MB en anglais)

2-a Entiers

- Conversion décimal -> binaire
  - Pour changer un nombre entier positif de base 10 en base 2
    - Si le nombre est pair, mettre 0, sinon mettre 1 à gauche du nombre binaire
    - Le diviser par 2 (division entiere) et recommencer
- Conversion binaire -> décimal
  - Pour changer un nombre entier positif en base 10 en base 2
    - Faire la somme des puissances de 2 représentées par le nombre

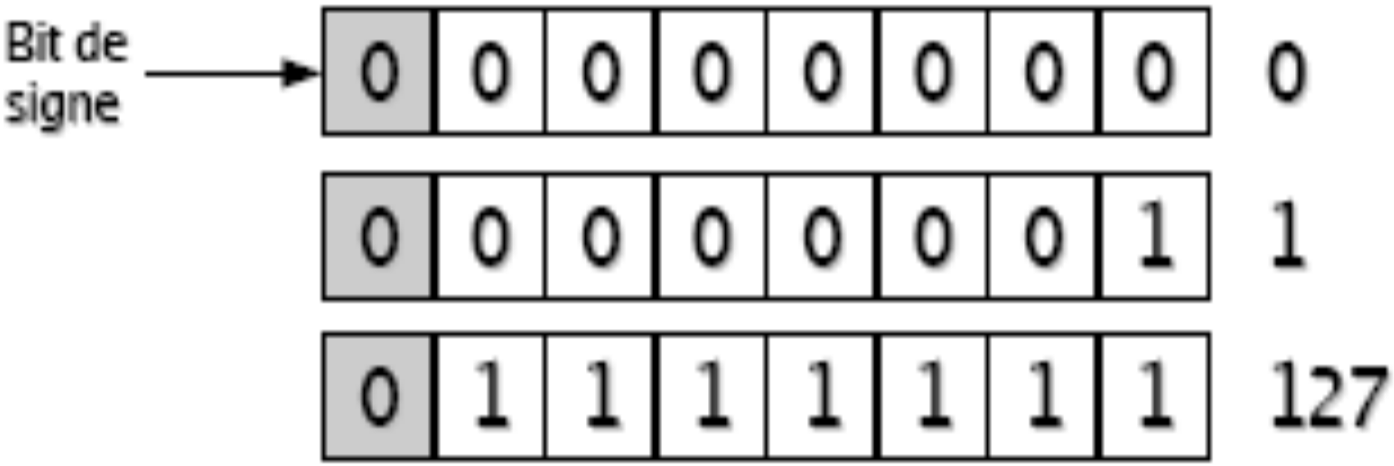
2-a Entiers

- Coder en binaire 112, 43
- Quels nombres décimaux sont représentés par
  - 10001
  - 11010

Même principe en base 16 (hexadécimal). 1 lettre = 4bits.

2-b Entiers relatifs

- On suppose que les entiers sont sur n bits
- Idée simple: mettre un bit de signe
- Bit de signe à 0: entier positif de 0 à 2^{n-1}-1



2-b Entiers relatifs

- Bit de signe à 1: entier négatif

1	0	0	0	0	0	0	1	-1

- Problème:
  - Addition et soustraction n'ont pas le même algorithme
  - Zéro a deux représentations, 0 et -0:

0	0	0	0	0	0	0	0	+0
1	0	0	0	0	0	0	0	-0

2-b Entiers relatifs

- Solution ? On veut:

	0	0	0	0	0	0	0	1	1
+	1	.	.	.	.	.	.	.	-1
	0	0	0	0	0	0	0	0	0

On doit avoir:

1	1	1	1	1	1	1	1	-1

2-b Entiers relatifs

Complément à 1 d'un nombre binaire:

- Inverser tous les bits

10011101 => 01100010 = 10011101

Complément à 2 d'un nombre binaire

- Calculer son complément à 1
- Ajouter 1

10011101 + 1 = 01100011

2-b Entiers relatifs

Le complément à 2 est l'opposé.

-1 est 11111111, -2 est 11111110, ..., -127 est 10000001

Unique 0:

$\overline{00000000} + 1 = 11111111 + 1 = 00000000$

Somme des opposés nulle:

$10011101 + \overline{10011101} + 1 = 10011101 + 01100010 + 1$

$= 00000000$

Nombres vont de -128 à 127

2-b Entiers relatifs

L'addition fait aussi la soustraction

	0	0	1	0	1	1	1	0	46
+	1	1	0	0	1	0	1	1	-53
	1	1	1	1	1	0	0	1	-7

Quelques valeurs sur 8bits

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1
1	1	1	1	1	1	1	1	-1
0	1	1	1	1	1	1	1	127
1	0	0	0	0	0	0	0	-128

2-b Entiers relatifs

- ATTENTION
  - Codage fini des entiers, risque d'overflow

01111111 + 00000001 = 10000000

127 + 1 = -128

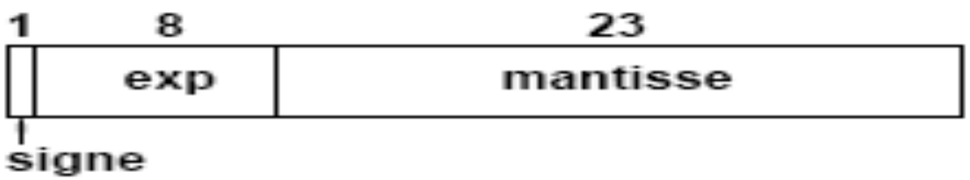
  - Le nombre est trop grand pour être représenté

2-c Représentation virgule flottante

Représentation normalisée IEEE 754

3 formats: 32, 64 et 80 bits (ce dernier que pour des calculs intermédiaires)

Pour représenter le nombre 1,m.2^{e} en 32 bits



signe est le signe du nombre (1: négatif, 0: positif)

- *exp* est un entier positif, avec *exp = e+127*
- *mantisse* est un entier positif, *mantisse = m*

2-c Représentation virgule flottante

- Les champs exposants 0000000 et 1111111 ont un sens particulier
- $1{,}0 = 1{,}0.\ 2^0$

mantisse nulle, exposant de 0+127=127:

0 01111111 00000000000000000000000

- $-18{,}625 = -(2^4 + 2^1 + 2^{-1} + 2^{-3}) = -(1 + 2^{-3} + 2^{-5} + 2^{-7})\ .\ 2^4$

1 10000011 00101010000000000000000

La mantisse est la suite $\alpha_1 \ldots \alpha_{23}$ correspondant à $\sum_{i=1}^{23} \alpha_i 2^{-i}$

2-c Représentation virgule flottante

- Représentation de 0
  - Mantisse à 0, champ exposant à 0 (signe 0 ou 1)
- Représentation de l'infini
  - Mantisse à 0, champ exposant à 11111111
  - Utilisé pour la division d'un nombre par 0.
- Représentation de NaN, Not a Number
  - Mantisse non nulle, champ exposant à 11111111
  - Utilisé pour le résultat de certaines opérations comme la division de l'infini par l'infini

2-c Représentation virgule flottante

Risque d'overflow: la valeur est trop grande pour être représentée.

- Plus grande valeur possible pour le champ de l'exposant: 11111110 c'est à dire un exposant 254-127 = 127

Risque d'underflow: perte possible de précision si opération entre opérandes d'ordre de grandeur très différents.

- Plus petite valeur possible pour le champ exposant: 00000001 soit un exposant 1-127 = -126

2-b Entiers relatifs

Exercice:

- Convertir 67, 42, -10, -88 en binaire et complément à 2 sur 8 bits
- Combien vaut 10010100 (complément à 2 sur 8 bits) en décimal ?
- Si tous les nombres signés sur 8 bits ont un inverse, et 0 n'a qu'une représentation, ca fait un nombre impair de nombres ! (255) Or sur 8 bits, il y a 256 valeurs possibles...Qui est l'intrus ?