Livre de référence : Architecture de l'ordinateur, Andrew Tanenbaum, 5ème édition

Plan

- L'architecture en couches
  - Introduction
- L'ordinateur et ses langages
  - Détail des couches
- L'exemple de Pascal
  - La multiplication sur le LMC
- Bref historique des machines à couche
- En quelques mots

Chacun son langage

- Le Pharaon :
  - « Je veux une pyramide »
- L'architecte :
  - « Il faut faire des fondations selon ces plans », « … », « … »
- Le chef de chantier :
  - « Creusez un trou de 20 pieds à l'intérieur du carré délimité par ces repères » , « … », « … »
- Le sous-chef :
  - Prenez une pelle et creusez là, là et là jusqu'à ce que je vous dise d'arrêter, « … », « … »
- L'ouvrier :
  - Sans le savoir, il va envoyer des influx nerveux à ses muscles en fonction de l'ordre reçu et de ses sens (le toucher et la vue principalement).

La traduction

- On suppose que chaque ordre dans une couche correspond à une séquence d'ordres dans la couche inférieure.
  - Ex : « je veux une pyramide » devient :
    - ✓« Faire un plan »
    - ✓« Faire les fondations »
    - ✓« Aller chercher les matériaux »
    - ✓« Assembler les matériaux selon le plan »
    - ✓« Vérifier la qualité de la construction »
- Il suffit alors de connaître les équivalences pour traduire chaque couche dans la couche inférieure et pouvoir donner à l'ouvrier son plan de travail pour 20 ans.

L'interprétation

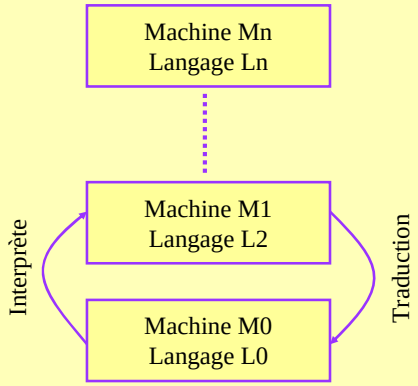
- A chaque niveau, on engage une personne capable de « traduire » (ou plutôt d' « interpréter ») les directives du niveau supérieur.
  - ✓L'ouvrier va donc demander à l'interprète du sous-chef ce qu'il doit faire.
  - ✓Celui-ci va faire de même avec l'interprète du chef de chantier
  - ✓Lui-même va devoir demander à l'interprète de l'architecte ce qu'il doit faire.
  - ✓Ce dernier va finalement s'adresser à l'interprète du pharaon pour savoir ce qu'il attend de lui.

La traduction comparée à l'interprétation

- La traduction
  - o Un gros travail au début
  - o Un gros « programme » pour l'ouvrier
  - o Très rapide car on ne pose plus de question
- L'interprétation
  - o Pas de traduction à faire au début
  - o Des ordres élémentaires à chaque niveau
  - o Il faut constamment faire appel à des interprètes
  - o L'information doit monter et redescendre ensuite toute la hiérarchie avant l'exécution

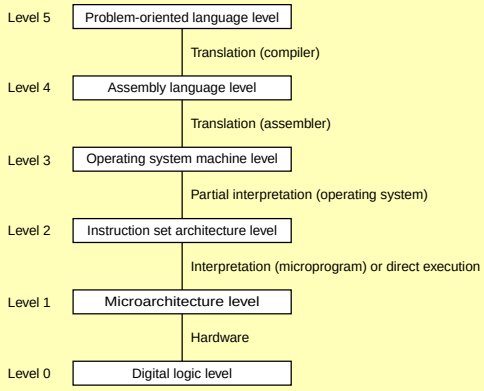
L'ordinateur et ses langages

- Un certain travail T sur une machine Mn (avec son langage Ln) peut être réalisé sur une machine Mn-1 (avec un langage Ln-1) si elle dispose d'un interprète ou si le programme a préalablement été traduit.



L'ordinateur et ses langages

- 5 : Langage de haut niveau
- 4 : Langage d'assemblage
- 3 : Système d'exploitation
- 2 : le jeu d'instructions
- 1 : micro-architecture
- 0 : Logique numérique



Quelques définitions

- Programme
  - Séquence d'instructions qui ont un sens dans le langage considéré
- Couche / Niveau
  - Une machine et le langage qui lui est propre
- Machine virtuelle
  - Machine capable d'exécuter un programme dans écrit dans son langage. Synonyme de « couche » ou encore « niveau »

Détail des couches (1/7)

- 0 La couche « logique numérique »
  - Toutes les opérations son effectuées en parallèle par des portes logiques élémentaires et séquencés par une horloge.
  - Couche supérieure : micro architecture
  - Couche inférieure : les transistors

Détail des couches (2/7)

- 1 La couche « microarchitecture »
  - Des fonctions logiques évoluées sont interconnectées :
    - ✓Registres
    - ✓Contrôleur
    - ✓Unité Arithmétique et logique
    - ✓…
  - Couche supérieures : ISA (jeu d'instructions)
  - Couche inférieure : Logique numérique

Détail des couches (3/7)

- 2 La couche ISA « jeu d'instructions »
  - Un processeur est capable d'exécuter un programme écrit en langage machine … une succession de « 0 » et de « 1 » qui codent les instructions.
  - La liste de toutes les instructions exécutable par un processeur constitue son « jeu d'instruction »
  - Couche supérieure : Système d'exploitation
  - Couche inférieure : Microarchitecture

Détail des couches (4/7)

- 3 La couche « Système d'exploitation »
  - De nombreuses fonctions sont utilisées par tous les programmes (lire une touche, afficher un caractère …)
  - Le système d'exploitation offre un ensemble de fonctions partagées par tous les programmes et leur permet aussi de travailler ensemble
  - Couche supérieure : langage d'assemblage
  - Couche inférieure : jeu d'instruction

Détail des couches (5/7)

- 4 La couche « Langage d'assemblage »
  - Un langage plus « convivial », directement utilisé par le programmeur d'applications pour accéder aux couches 1,2 et 3. Il est constitué de mots « mnémoniques » pour éviter au programmeur d'écrire des « 0 » et des « 1 ». Cela reste un langage de très bas niveau.
  - Couche supérieure « Langages d'application »
  - Couche inférieure « Systèmes d'exploitation »

Détail des couches (6/7)

- 5 La couche « Langage de haut niveau»
  - Ce sont plus répandus (des centaines). Ils permettent aux programmeurs d'écrire des applications dans un formalisme plus complexe et structuré. Cela permet de valider un programme de façon plus rapide et avec moins de bogues. Exemples : Pascal – Java – C – C++
  - Niveau supérieur : les générateurs de code …
  - Niveau inférieur « Langages d'assemblage »

Détail des couches (7/7)

- « 6 » La couche « Générateur de code»
  - Certaines applications génèrent du code de haut niveau qui sera ensuite utilisé par des programmeurs d'application ou bien seront intégrés directement à un logiciel.
  - Exemple : Le simulateur de circuits logique
    - ✓Traduction du circuit en code C
    - ✓Compilation du C en langage d'assemblage
    - ✓Compilation du langage d'assemblage en langage machine
    - ✓Exécution de la simulation et affichage des résultats.

Exemple : de la machine à calculer

- Pascal dit « Je veux une machine capable de réaliser des multiplications »
  - « Quelle est la taille des nombres à multiplier ? »
    - ✓ Réponse : « 8 bits »
  - « Comment entre-t-on les nombres ? »
    - ✓ Réponse : « au clavier, en base 10 »
  - « Comment retourne-t-on les résultats ? »
    - ✓ Réponse : « à l'écran, en base 10 »
  - Que faut-il faire en cas d'erreur d'entrée ?
    - ✓ Afficher un message d'erreur et arrêter tout
  - Que faut-il faire en cas de dépassement ?
    - ✓ Afficher un message d'erreur et arrêter tout
  - Doit-on traiter aussi les nombres négatifs ?
    - ✓ Non

Exemple : de la machine à calculer

- Le programmeur d'application écrit un code proche de ceci dans un langage de haut niveau :

```
int a,b,c;
while (true) {
   scanf(«%i», &a);
   scanf(«%i», &b);
   if ((a<0) | (a>255)) {printf(« Dépassement »); return -1;}
   if ((b<0) | (b>255)) {printf(« Dépassement »); return -1;}
   c=a*b;
   printf(« Le produit de %i par %i est : %i», a, b, c »);
}
```


Exemple : de la machine à calculer

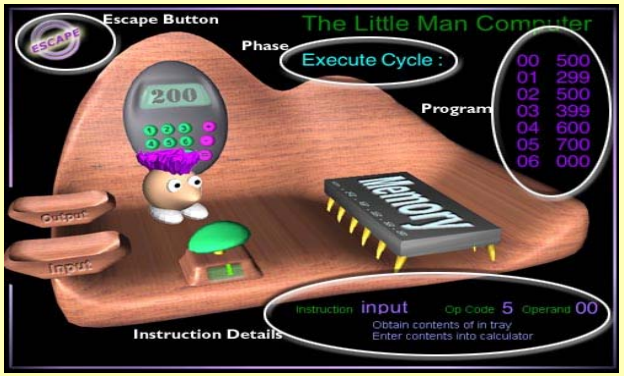
- Le compilateur traduit le code comme ceci :

```
int a,b,c;                  Réserve de l'espace en mémoire
while (true) {              Traduit en langage d'assemblage
   scanf(«%i», &a);         Appel système
   scanf(«%i», &b);         Appel système
   if ((a<0) | (a>255)) …   Traduit en langage d'assemblage
   if ((b<0) | (b>255)) …   Traduit en langage d'assemblage
   c=a*b;                   Traduit en langage d'assemblage
   printf(« Le produit …    Appel système
}
```


Exemple : de la machine à calculer

- Dès que l'on quitte le haut niveau, les étapes suivantes doivent tenir compte du type des couches inférieures.
- Dans notre exemple, nous travaillons avec une machine virtuelle LMC sur laquelle tourne un système d'exploitation imaginaire.

Little Man Computer (LMC)

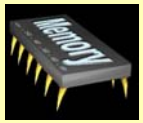


http://www.herts.ac.uk/ltdu/projects/mm5/

LMC, la couche microarchitecture (1/4)

- Le chef : « Little Computer Man » : Il ne sait rien faire par lui-même mais il sait ce qu'il faut faire et il a autorité sur tous les autres.

- Le compteur de programme « Program Counter » : Il indique le numéro du tiroir de la prochaine instruction à réaliser.



- La mémoire : une armoire à tiroirs numérotés de 0 à 99. Chaque tiroir peut contenir un nombre de 0 à 999.

LMC, la couche microarchitecture (2/4)

- L'unité de calcul. Elle est capable de faire des additions et des soustractions. Elle dispose d'une mémoire interne qui garde toujours le résultat de la dernière opération.

- Les entrées/sorties. Un ordinateur ne sert à rien s'il est incapable de communiquer avec le monde extérieur. Le LMC est capable de recevoir et d'envoyer des nombre de 0 à 999

LMC, la couche microarchitecture (3/4)

Le LCM effectue toujours le même cycle:

- Lire le compteur de programme pour savoir dans quel tiroir se trouvent les chiffrent qui codent l'instruction à exécuter.
- Incrémenter (faire +1) le compteur de programme.
- Lire les chiffres qui se trouvent dans le tiroir concerné. (FETCH)
- Regarder de quelle instruction il s'agit (DECODE)
- Exécuter l'instruction en question (Execute)

On se rapproche déjà de la couche logique …

LMC, la couche microarchitecture (4/4)

- 100 tiroirs numérotés de 0 à 99 pouvant contenir des nombres de 0 à 999.
- Un nombre de 0 à 999 est en fait 3 chiffres de 0 à 9

3	1	7

- On peut donc voir la mémoire comme 100 tiroirs contenant chacun trois compartiments. Dans chaque compartiment, on peut mettre un chiffre de 0 à 9. En fonction du contexte, ces chiffres peuvent avoir différentes significations

A l'intérieur du LMC, la couche ISA

- Dans le LMC, il y a trois formats de données:
  - Le format « numérique »
    - ✓ Le premier chiffre représente les centaines, le deuxième représente les dizaines et le troisième représente les unités.
  - Le format « instruction - tiroir»
    - ✓ Le premier chiffre représente le type d'instruction.
    - ✓ Les deuxième et troisième chiffres représentent un numéro de tiroir (de 0 à 99).
  - Le format « instruction spéciale»
    - ✓ Quand le premier tiroir contient le chiffre 8, il s'agit d'une instruction spéciale déterminée par les deux tiroirs suivants.

A l'intérieur du LMC, la couche ISA

1NN	Load	Copier la valeur du tiroir dans l'unité de calcul
2NN	Store	Copier la valeur de l'unité de calcul dans le tiroir
3NN	Add	Ajouter le contenu du tiroir à celui de l'unité de calcul
4NN	Subtract	Soustraire le contenu du tiroir de celui de l'unité de calcul
5NN	Input	Aller chercher une donnée d'entrée et la placer dans l'unité de calcul
6NN	Output	Copier la donnée de l'unité de calcul et l'envoyer en sortie
7NN	Halt	On arrête tout
800	Skip If Negative	Ne pas exécuter l'instruction suivante si le résultat est négatif
801	Skip If Zero	Ne pas exécuter l'instruction suivante si le résultat est nul
802	Skip If Positive	Ne pas exécuter l'instruction suivante si le résultat est positif (ou nul)
9NN	Jump	Ecrire la valeur du tiroir dans le compteur d'adresse

Exemple simple introductif (1/3)

« Réalisez un programme qui prend deux nombre en entrée et retourne la somme en sortie »

1) Lire la première entrée (*Input)*
2) Lire la deuxième entrée (*Input*)
3) Additionner les deux (*Add*)
4) Envoyer le résultat (*Output*)

Problèmes :

Input écrit la valeur dans l'unité de calcul et il n'y a qu'une place.

Add calcule la somme du contenu d'un tiroir avec le contenu courant de l'unité de calcul.

Exemple simple introductif (2/3)

Il faut donc mémoriser temporairement la première donnée lue.

Le programme devient:

1) Lire la première entrée (*Input)*
2) Ecrire cette donnée en mémoire
3) Lire la deuxième entrée (*Input*)
4) Additionner le contenu de l'unité de calcul avec le contenu du tiroir (*Add*)
5) Envoyer le résultat (*Output*)

Quel tiroir utiliser pour sauvegarder la donnée ?

Exemple simple introductif (3/3)

Le contenu de l'armoire à tiroirs s'écrit :

N° tiroir	Instruction	Contenu du tiroir		
0	INPUT	5	X	X
1	STORE 99	2	9	9
2	INPUT	5	X	X
3	ADD 99	3	9	9
4	OUTPUT	6	X	X
5	HALT	7	X	X
…	…	X	X	X
99	réservé	X	X	X

Le cas de la multiplication (1/3)

« Réalisez un programme qui prend deux nombres en entrée et retourne le produit en sortie »

1) Lire la première entrée (*Input)*
2) Ecrire cette donnée en mémoire
3) Lire la deuxième entrée (*Input*)
4) Multiplier les deux (???)
5) Envoyer le résultat (*Output*)

Problème : l'instruction multiplication n'existe pas !

Le cas de la multiplication (2/3)

A*B = B + (A-1)*B

En terme de fonction : Mult(A,B) = B+Mult(A-1,B)

or, Mult(0,B) est trivial.

Par récursivité, on peut donc multiplier deux nombres rien qu'en faisant des additions.

Introduisons des variables auxiliaires :

X et Z de sorte que Z+X*B=A*B à tout moment.

Au début, Z=0 et X=A

Il suffit alors de répéter $\{Z_{new}=Z+B; X_{new}=X-1\}$ jusqu'à ce que X soit égal à 0.

Le cas de la multiplication (3/3)

Dans l'exemple de Pascal, la seule différence vient du fait que les opérandes sont déjà placées en mémoire par les appels systèmes. Il n'est donc plus nécessaire de faire les opérations d'entrées/sorties dans le code :

0) Lire B (tiroir 47 pour B) // pour la boucle
1) Ajouter B à Z (tiroir 46 pour Z)
2) Ecrire le résultat dans Z (tiroir 46 pour Z)
3) Lire X (tiroir 48 pour X)
4) Soustraire la constante 1 (tiroir 45 pour la cste 1)
5) Ecrire le résultat dans X (tiroir 48)
6) Si X=0, passer l'instruction suivante
7) Aller à l'opération 0
8) HALT

La multiplication en langage d'assemblage

(tiroir 45 pour la cste 1) (tiroir 46 pour Z)
(tiroir 47 pour B) (tiroir 48 pour X)

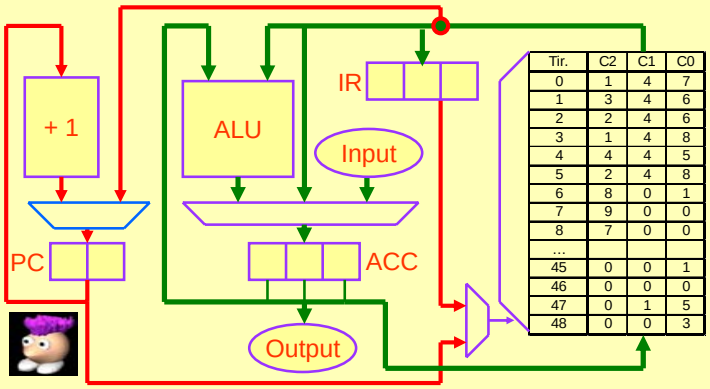
```
0) LOAD   47   ; Lire B
1) ADD    46   ; Ajouter B à Z
2) STORE  46   ; Ecrire le résultat dans Z (tiroir 46 pour Z)
3) LOAD   48   ; Lire X
4) SUB    45   ; Soustraire 1
5) STORE  48   ; Ecrire le résultat dans X
6) SKIP   Z    ; Si X=0, passer l'instruction suivante
7) JUMP   0    ; Aller à l'opération 0
8) HALT        ; Arrêter
```


La traduction en langage machine

Le langage machine du LMC est écrit en base 10 (cas très rare si pas unique). Chaque instruction assembleur doit être traduite dans son format décimal avant de pouvoir être exécutée par le LMC :

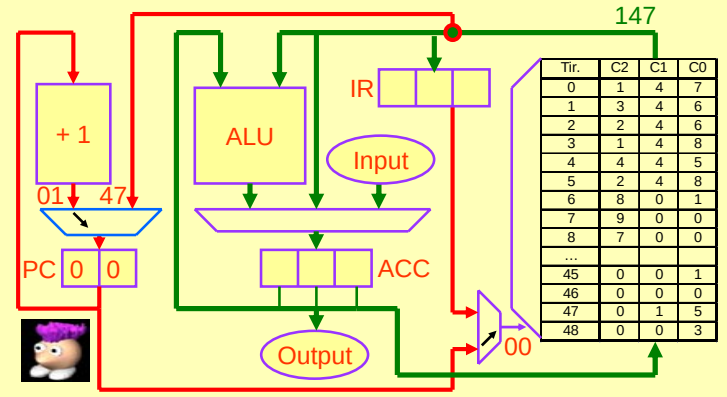
```
0) LOAD   47   ;   = 147
1) ADD    46   ;   = 346
2) STORE  46   ;   = 246
3) LOAD   48   ;   = 148
4) SUB    45   ;   = 445
5) STORE  48   ;   = 248
6) SKIP   Z    ;   = 801
7) JUMP   0    ;   = 900
8) HALT        ;   = 700
```


LMC, la couche microarchitecture : démo

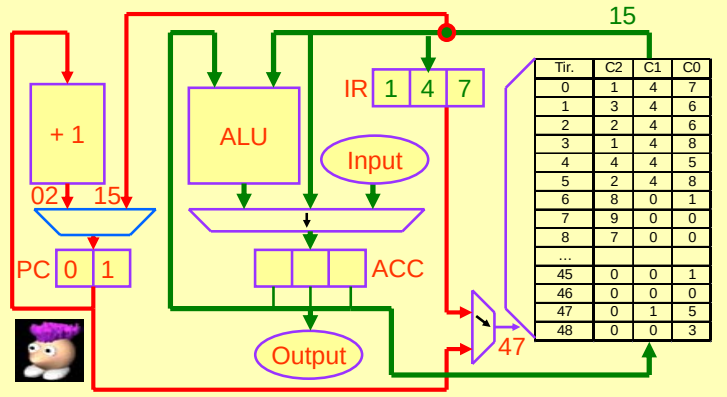


Tir.	C2	C1	C0
0	1	4	7
1	3	4	6
2	2	4	6
3	1	4	8
4	4	4	5
5	2	4	8
6	8	0	1
7	9	0	0
8	7	0	0
...			
45	0	0	1
46	0	0	0
47	0	1	5
48	0	0	3

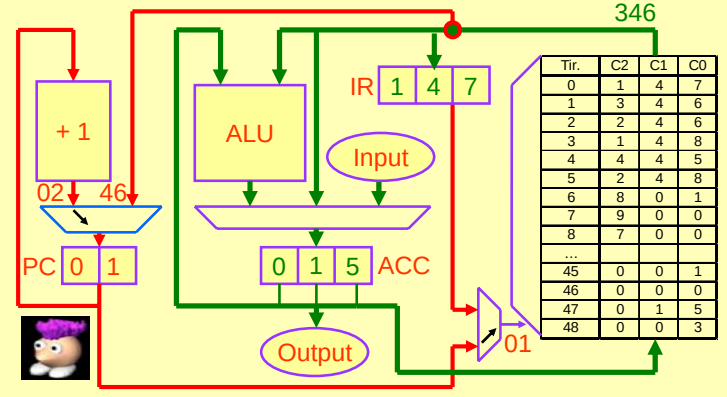
LMC, la couche microarchitecture : démo



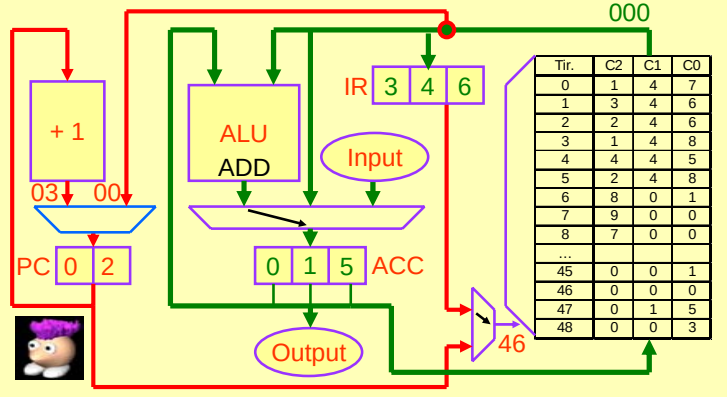
LMC, la couche microarchitecture : démo



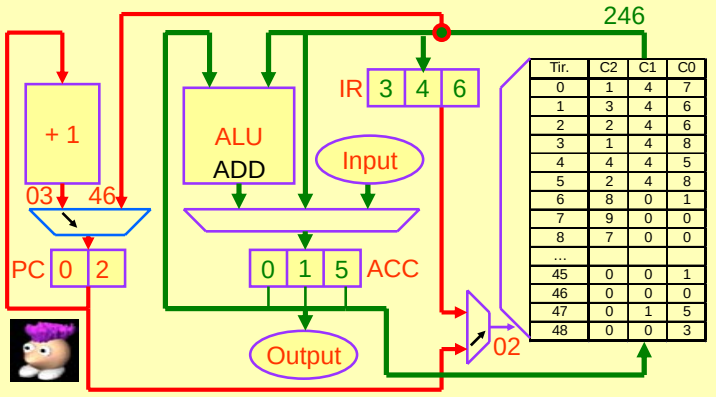
LMC, la couche microarchitecture : démo



LMC, la couche microarchitecture : démo



LMC, la couche microarchitecture : démo



L'arithmétique signée en base 10 (1/2)

- 100 + 999 = 1099
  Sur le LMC, ce résultat devient 099
- 250 + 999 = 1249
  Sur le LMC, ce résultat devient 249

- En fait, +999 revient à faire -1 !!!

…

999 = 1000-1

X+999 = X-1 + 1000

Or, le LMC retire le chiffre des milliers. Donc, si X>=1, le résultat dépasse 1000. Le LMC retire donc 1000 et on se retrouve avec X-1 !!!

L'Arithmétique signée en base 10 (2/2)

- De manière générale, soustraire Y revient à ajouter 1000-Y

- Exemples : -45 devient 955
  300 + (-45) = 300+955 = 1255
  et le LMC ne garde que 255

Il faut une convention pour fixer les choses. Par exemple : tous les nombres >=500 sont considérés comme étant négatifs.

Si on a besoin de plus de valeurs, il faut utiliser plusieurs tiroirs pour mémoriser un nombre. Avec deux tiroirs par exemple, on a droit à 6 chiffres et donc des valeurs comprises entre -500000 et +499999 qui seront représentés par des nombres entre 0 et 999999.

Le matériel et le logiciel

- La partie physique, électronique d'un dispositif constitue le matériel.
- La partie algorithmique, programmation, constitue le logiciel.

Matériel et Logiciel sont logiquement équivalents

- A postériori, la différence est évidente mais à priori, la frontière est floue. Tout problème peut être résolu par l'un ou l'autre ou une combinaison des deux. Exemple : la multiplication

L'invention de la microprogrammation

- Le processeur contient lui-même un microprogramme qui lui permet d'interpréter le programme machine réalisé par le programmeur d'applications.
- Le jeu d'instruction dépend du microprogramme et peut être modifié en fonction des besoins
- Le matériel est considérable réduit malgré que le jeu d'instructions est complexe

L'invention des systèmes d'exploitation

- Développés au départ pour réduire les manipulations avec les cartes perforées …
- Offre un partage des ressources pour plusieurs applications et/ou plusieurs utilisateurs.
- Permet d'assurer la compatibilité des logiciels quand le matériel évolue.

La migration des fonctionnalités vers le microcode

- De nouvelles instructions apparaissent :
  - L'incrémentation (+1)
  - Multiplication et division entière
  - Calcul en virgule flottante
  - Appel de procédure et retour
  - Accélération de boucles
  - Manipulation de chaînes de caractères
  - Manipulation de vecteurs
- Et aussi des concepts plus évolués :
  - Manipulation de tableaux
  - Possibilité de relocaliser un programme en mémoire
  - Les interruptions
  - La commutation

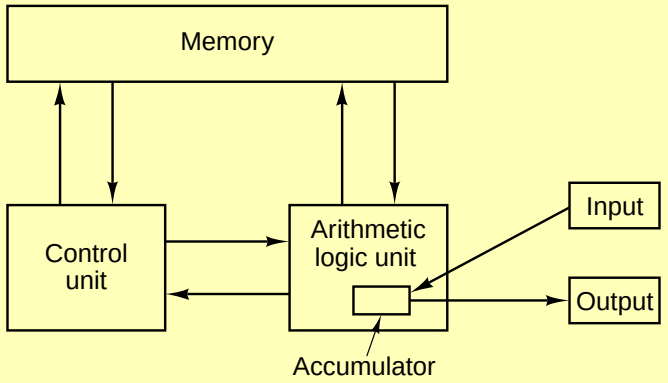
Disparition de la microprogrammation

- Les microprogrammes étant de plus en plus longs et lents pour interpréter le langage machine, une nouvelle architecture apparaît : le RISC.
- Le principe est d'exécuter une instruction élémentaire à chaque cycle d'horloge plutôt que d'avoir un code complexe à interpréter

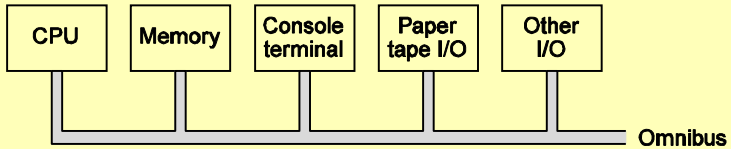
Les grandes étapes de l'architecture des ordinateurs

- Les calculateurs mécaniques (1642 – 1945)
- Les tubes à vides (1945 – 1955)
  - L'architecture Von Neumann
- Les transistors (1955-1965)
  - L'Architecture à bus
- Les circuits intégrés (1965 – 1980)
  - SSI – MSI – LSI
- Le VLSI (1980 - …)
  - La loi de Moore
- À quand l'ordinateur quantique ?

L'architecture Von Neumann



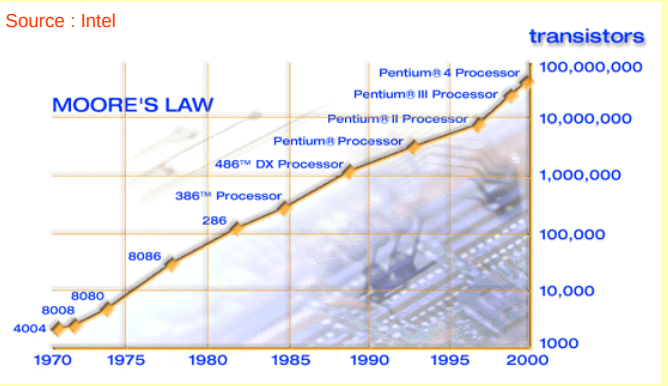
L'architecture Bus



- Tous les composants de l'ordinateur partagent les mêmes lignes électriques pour s'échanger des données
- Un composant à la fois est le maître du bus et peut négocier un transfert de données entre les composants
- Chaque composant a son adresse propre

La loi de Moore

Source : Intel



En quelques mots

- Programme — Compilateurs
- Traduction — Interprétation
- Machine virtuelle — Couches / Niveaux
- Registres — Niveau microarchitecture
- UAL — Chemin de données
- Microprogramme — Langage Machine
- Couche 0 : composant
- Couche 1 : logique numérique
- Couche 2 : architecture du jeu d'instructions
- Couche 3 : système d'exploitation
- Couche 4 : langage d'assemblage – assembleur
- Couche 5 : langages d'application (de haut niveau)
- Matériel/Logiciel logiquement équivalents
- Appel au système
- Proposition de Von Neumann — Loi de Moore