Architecture matérielle et logicielle

Contents

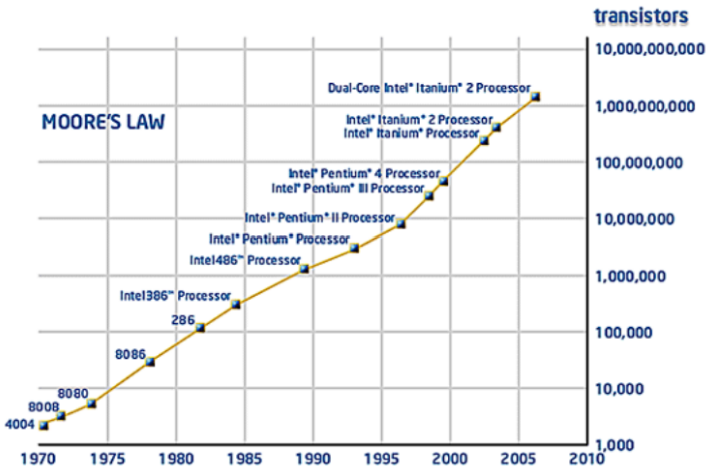
Part I
Le Cours

1 Généralités

L'informatique est la science du traitement de l'information et ayant pour outil l'ordinateur. On peut mesurer les performances d'un ordinateur suivant plusieurs paramètres, comme par exemple la période horloge (qu'on trouve grâce à la fréquence processeur). Ce paramètre définit la rapidité à laquelle l'ordinateur est capable de faire tourner un programme.

2 Micro-processeur

Loi de Moore :



La loi de Moore définit l'évolution temporelles des composants électroniques.

Le micro-processeur est un circuit intégré numérique programmable capable d'exécuter des actions en fonction d'un programme externe.

Un cycle de micro-processeur se décompose ainsi :

- Fetch : recherche du code opératoire de l'instruction
- Decode : décodage et séquencement de l'instruction
- Execute : exécution de l'instruction
- Write Back : stockage du résultat si nécessaire

3 Mémoire

L'accès en mémoire est un facteur de performance d'un ordinateur. Les différents types de mémoire sont :

- ROM : Read Only Memory, mémoire non volatiles
- DRAM : Dynamic RAM
- SRAM : Static RAM, plus rapide que DRAM

Les mémoires utilisées actuellement sont SDRAM, la mémoire cache contient un peu de mémoire centrale en mémoire d'accès rapide.

4 Performances

Pour mesurer la performance liée à l'exécution d'un programme :
Durée totale d'exécution d'un programme = Nombre et type d'instructions × Nombres de périodes × Période d'horloge

On peut donc agir sur les programmes (instructions), l'architecture et la technologie.
Pour améliorer les performances du processeur, on mise actuellement sur :

- pipelining : exécuter une nouvelle instruction avant que l'ancienne soit finie
- superscalaire : possibilité de traitement en parallèles de plusieurs instructions, il faut faire attention au dépendance de celle-ci
- hyperthreading : plusieurs processeurs logiques avec leurs propres registres et compteur ordinaux sont intégrés
- multicoeurs : plusieurs processeurs sur la même puce, diminue la dissipation de la chaleur

Les caractéristiques d'un composant mémoire sont temps d'accès, volatilité et capacité.

Part II
TPs

1 Création d'un exécutable :

- préprocesseur : recherche et utilise les fichiers en-tête (.h) pour le compilateur, et produit un fichier.i . Il effectue la substitution de texte (remplace les constantes), la compilation conditionnelle, et l'inclusion des fichiers.h
- compilateur : détecte les erreurs de syntaxe, traduit le code source en code machine, crée un fichier.o . Il analyse la syntaxe, crée une table des symboles (définit les adresses des constantes) et crée un fichier.o
- éditeur de liens : produit le fichier exécutable. Il compile séparément les fichiers, édite des liens entre ces fichiers. Il concatène les fichiers.o, résout les références extérieures, transforme les adresses, il utilise le format ELF (Executable and Linkable Format)

2 Les bibliothèques

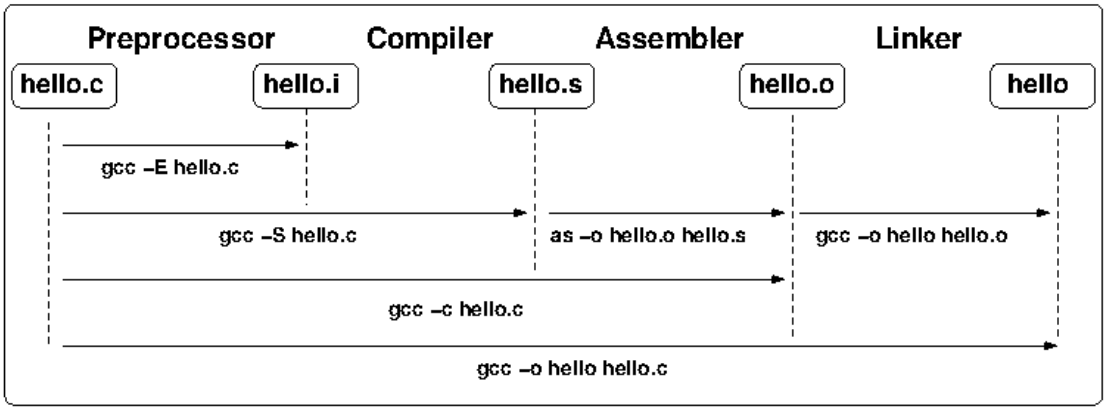
Bibliothèques statiques : libxxxxx.a

- regroupe des fichiers.o
- gcc : -l<NOM> -L<LIEN>
- avantage : tout le code est prêt pour l'éxecution
- inconvénients : taille, duplication en mémoire

bibliothèques dynamiques : xxxxx.so

- contient des références à d'autres codes
- gcc : -l<NOM> -L<LIEN>

Make
voir pougne algo



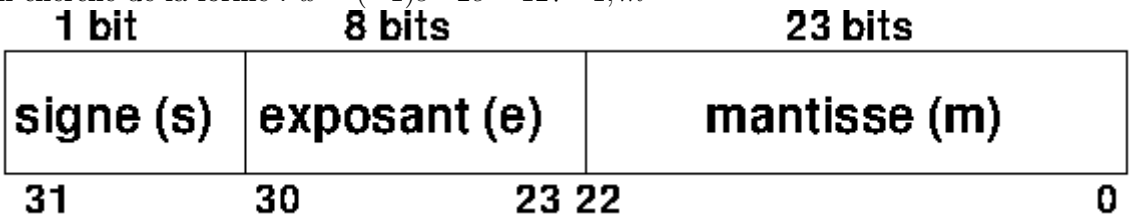
3 Codage de l'information

Les informations sont codés suivant leurs natures et suivant plusieurs codages. Pour les caractères, on peut citer le codage ASCII, ISO 8859-1 ou l'Unicode.

Passer d'un nombre réel à son codage en hexadécimal On va coder -13,375 (exemple du cours)

$$\begin{aligned} -13,375 &= 8+4+1+0,25+0,125 && \text{on décompose en base 2}\\ &= 23+22+20+2-2+2-3 &&\\ &= 1101,011 && \text{transformation en binaire}\\ &= 1,101011*23 && \end{aligned}$$

On cherche de la forme : $x = (-1)s * 2e - 127 * 1, m$



On a donc 1 pour le signe négatif (s), 101011 pour la mantisse (on ne prend pas le premier 1, m), et 127+3=130 comme exposant (e) Donc on a 1 10000010 10101100000000000000000 Soit 1100 0001 0101 0110 0000 0000 0000 0000 Ce qui donne les nombres hexadécimaux C 1 5 6 0 0 0 0

Le processus inverse fonctionne de même.

4 Assembleur

Les fichiers en Assembleur sont les fichiers.s, ils sont transformés en fichiers.o avec l'appel de l'assembleur as. L'assembleur est un langage de bas niveau, il est efficace et donne accès au matériel, mais le coût de développement en Assembleur est important.

Dans une instruction assembleur, qui correspond à une instruction processeur, on retrouve le schéma suivant : Étiquette | Mnémonique | opérande(s) | commentaire

- L'étiquette est facultative et désigne un emplacement mémoire
- Le Mnémonique est un mot réservé du langage
- Les opérandes sont les objets concernés par le mot mnémonique
- Le commentaire est facultative et est une note du codeur

Types d'instructions :

- instructions arithmétiques : add (addition), sub (soustraction), inc (incrémentation), dec (décrémentation), cmp (comparaison), mul (multiplication), div (division)
- instructions logiques : not, and, or, xor

- instructions de branchement : call (appel à une fonction), ret (retour), jmp (saut jusqu'à une étiquette)

Adressage en Assembleur

- adressage par registre : movq %rsp, %rbp; on met la valeur de rsp dans rbp
- adressage immédiat : movb $1, %al; on met la valeur 1 dans al
- adressage direct : cmpl nombre, %eax; on met la valeur de la variable globale nombre dans eax
- adressage implicite : clc; on remet à zéro le bit de carry, l'adresse est comprise dans l'instruction
- adressage indirect : movl $3,(%rax); on met 3 dans le mot pointé par rax, rax contenant une adresse
- adressage basé/indexé : deplacement(base,index,echelle); l'adresse est base + déplacement + index*échelle

5 Pile

La pile est dans le registre rsp, les données sont stockées ici et pour y accéder, on utilise rbp

6 Processus

définition programme en cours d'exécution, caractérisé par un numéro unique (pid), une entrée dans le pseudo système de fichiers (/proc), et un processus père
On utilise ps pour en faire la liste, pstree pour les représenter en arbre.
Il existe deux types de processus, les processus utilisateurs et les processus démons (qui sont lancés au démarrage).
Un processus a besoin de différentes ressources : processeur, mémoire, et entrées-sorties. Mais il doit les partager avec les autres processus.

- Mode noyau : exécution d'instructions privilégiées
- Exclusion mutuelle : pas d'interférences entre les processus, attente de libération
- commutation de processus : On change de processus en cours de tâche, on sauvegarde le contexte (valeur des registres, pile, ...)
- Ordonnancement : on a une file d'attentes, avec un mécanisme de priorité (FIFO, Shortest Job First, Round Robin : tourniquet)

Clonage de processus On appelle fork, il crée deux processus pour la suite du programme. Il renvoie 0 pour le fils et le pid du fils au père.

Recouvrement On utilise les fonctions exec{l|v}[e|p], on abandonne le programme en cours et on poursuit avec le processus donné en paramètre

- l : liste
- v : tableau
- e : environnement
- p : varible PATH

exemple : execlp(Chemin_du_programme,liste_arguments_terminée_par_NULL)
execvp(Chemin_du_programme, tableau_arguments)

Terminaison exit
Pour attendre, on utilise wait(&status) où status est un entier qui passe à 0 après l'exécution du fils