PRESENTATION

Intitulé : Architecture des Ordinateurs I

Coefficient : 5

Objectif du module : Permettre à l'étudiant d'appréhender l'organisation et le fonctionnement d'un ordinateur indépendamment des aspects réalisation et technologie.

Contenu simplifié du module:

Ch1 : Introduction générale à l'architecture des ordinateurs (10 %)
Ch2 : Architecture de VON NEUMANN (30 %)
Ch3 : Représentation des informations de base (15 %)
Ch4 : Langage machine (5 %)
Ch5 : Répertoire d'instructions d'une machine (10 %)
Ch6 : Modes d'adressage (5 %)
Ch7 : Utilitaires de base (5 %)
Ch8 : Sous-système d'entrées/sorties (20 %)

Bibliographie :

1-Architecture de l'ordinateur. Andrew tanenbaum (DUNOD 2001)
2-Architecture et technologie des ordinateurs. Paolo zanela (DUNOD 1999)
3-Structure des ordinateurs. Koudil (OPU 1994)
4-Architecture et fonctionnement des ordinateurs. Belaid (PAGES BLEUES 2001)
5-Principles of computer architecture. Murdocca (PRENTICE HALL 2000)
6-Guide pratique de programmation en assembleur. Benabdji (HOUMA 2000)
7-Architecture des systèmes informatiques. Ait Aoudia (OPU 1999)
8-The essentials of computer organization and architecture. Null (JONES BAR 2003)
9-Internet est une ressource importante .

Déroulement du module :

La présence aux séances de cours est nécessaire.
Cours magistral avec des polycopiés de synthèse établis par l'enseignant.
Des séries d'exercices seront remis aux étudiants avant les séances de TD pour préparations.
La présence et la participation aux séances des TD et TP sont obligatoire.
Les TP se composent de 2 parties (partie programmation et partie technologie).
En fin d'année, une note est attribuée à chaque étudiant en tenant compte de la préparation des TD, la participation aux TD et la présentation des TP .

INTRODUCTION GENERALE A L'ARCHITECTURE DES ORDINATEURS

Nb : (*les figures seront présentées pendant les séances de cours*)

DEFINITIONS :

L'Informatique : est la science de traitement des informations par des moyens automatiques.

Information : l'information est un fait qui possède un sens par opposition au terme donnée qui est abstraite. Par abus de langage on ne fait pas de différence entre les 2 termes. Au sens large, l'information apparaît comme le moyen de la communication. Tout ce qui peut se représenter, s'écrire et se dire pour être communiqué entre hommes et entres machine constitue de l'information.

Traitement : le traitement de l'information est le déroulement systématique d'une suite d'opérations sur des informations élémentaires .

Système : est un ensemble d'éléments matériels ou immatériels liés par des interactions (en état de marche, on a des changements d'état des éléments pour aboutir aux objectifs du système : notion d'entrées et sorties). Système solaire , système entreprise .....

Système informatique : un système informatique est un ensemble de matériels et logiciels destinés à réaliser des taches qui mettent en jeu le traitement automatique de l'information.

Le matériel (hardware) : désigne tout ce qui a un caractère matériel dans une machine.
(non modifiable)

Le logiciel (software) : Désigne tout ce qui n'est pas matériel (immatériel). Il est assimilé à la matière grise. Le logiciel est formé de la représentation informatique des algorithmes : les programmes. Un programme est une suite d'instructions exécutables par la machine. (modifiable)

Ordinateur : est une machine électronique de traitement de l'information. Cette machine est capable d'acquérir et de conserver des informations, d'effectuer des traitements et de restituer les informations sous le contrôle du logiciel.

Nb : un ordinateur sans logiciel est totalement inutile. Les notions de matériel et logiciels sont indissociables.

Architecture des ordinateurs: l'architecture des ordinateurs est la discipline qui correspond à la façon dont on conçoit les composants d'un système informatique.

L'architecture concerne les caractéristiques fonctionnelles d'un système informatique vu par le programmeur .

L'architecture des ordinateurs est en constante mutation vu les poussées technologiques : arithmétiques des ordinateurs -> jeu d'instructions -> conception des composants .....

Architecture = Jeu d'instructions + Organisation

Architecture matérielle : est la conception fonctionnelle des différents composants de la machine. (fonctionnement logique de chaque composant et le dialogue entre les composants)

Architecture logicielle : correspond au codage de l'information et au jeu d'instructions de la machine càd l'ensemble des opérations que la machine peut exécuter.

La Technologie : étude des composants : transistor , résistances ,…..

La Logique : assemblage des composants pour en faire des circuits et des unités

L'architecture est indépendante vis à vis de la réalisation et de la technologie.

Aujourd'hui , la conception de la machine suppose une collaboration très étroite entre les concepteurs du matériel et les concepteurs du logiciel.

MODELE EN COUCHES :

Pour l'étude et la conception de systèmes complexes , l'utilisation des concepts d'abstraction et de décomposition hiérarchique sont nécessaires pour une bonne maîtrise de cette complexité.

Abstraction : description simplifiée d'un système mettant l'emphase sur les aspects essentiels (Quoi ?) et ignorant certaines d'autres (Comment ?) càd la suppression de certains détails non significatifs et ainsi faciliter la compréhension d'un système pourtant complexe.

Approche hiérarchique : décomposition d'un système en plusieurs couches empilées où la couche supérieure utilise/communique avec la couche inférieure via une interface bien définie (Quoi ?) , La couche inférieure dissimulant ainsi de nombreux détails de mise en œuvre (Comment ?) à la couche supérieure.

Machine virtuelle : c'est une machine abstraite disposant d'un langage permettant aux usagers de communiquer avec ladite machine.

Une machine virtuelle définit un langage et un langage est définit par une machine virtuelle.

Si le langage est difficilement manipulable par l'être humain, on définit des langages de plus haut niveau facile à utiliser et ainsi de suite jusqu'à ce qu'on obtienne un jugé convenable.

Le langage du bas est le plus complexe

Le langage du haut est le plus simple

Le passage d'une machine (langage) à une autre se fait par interprétation ou traduction.

Interprétation : chaque instruction d'un programme écrite en langage L_n est traduite en la séquence d'instructions adéquate du langage L_{n-1} exécutables donc par la machine M_{n-1}

Traduction : un programme de la machine M_n est traduit une fois pour tout en un programme en langage L_{n-1} exécutable directement par la machine virtuelle M_{n-1}.

Avantage de l'abstraction : il est possible de maîtriser la complexité du système et disposer de méthodes de conception structurées ,en plus on peut modifier la structure interne d'une couche sans entraver le fonctionnement de l'ensemble.

STRUCTURE EN COUCHES D'UN SYSTEME INFORMATIQUE :

Les systèmes informatiques sont conçus comme une suite de niveaux où chaque niveau est bâti sur les fonctions de ces prédécesseurs.

Lorsqu'on conçoit un nouveau système informatique, il convient de choisir les instructions qui formeront son langage. Pour pouvoir réduire la complexité et le coût des circuits de la machine réelle, on opte généralement pour un langage simple. Le problème est que ces langages sont alors si primitifs qu'il est extrêmement pénible et fastidieux de les utiliser donc il faut construire un nouveau jeu d'instructions plus pratique à utiliser que le langage de base.

Afin de libérer l'utilisateur de la complexité du matériel et tous les détails de fonctionnement de l'ordinateur, il faut lui présenter une interface ou machine virtuelle plus facile à comprendre et à programmer.

Système en couche actuel :

Niveau 6	Couche applications
Niveau 5	Couche langage de haut niveau
Niveau 4	Couche langage assembleur
Niveau 3	Couche système d'exploitation
Niveau 2	Couche jeu instruction (langage machine)
Niveau 1	Couche microarchitecture
Niveau 0	Couche physique (logique numérique)

Evolution du système en couches :

- Initialement existence de 2 niveaux : niveau langage machine avec lequel on faisait les programmes et le niveau physique qui exécutent ces programmes (les années 40)
- En 1951 Wilkes proposa l'idée d'un système à 3 niveaux : adjonction de la couche microarchitecture entre les 2 autres niveaux
- Développement de l'assembleur dans les années 50
- Apparition du fortran (1957) langage de haut niveau
- Développement des systèmes d'exploitation au début des années 60

Les couches ou niveaux ne sont pas fixes et rigides : matériel et logiciel sont équivalents

A chaque niveau correspondent des utilisateurs potentiels.

ARCHITECTURE DES MACHINES REELLES :

Organisation de base d'un ordinateur : un ordinateur comprend les unités fonctionnelles suivantes :
Processeur : cerveau de l'ordinateur, supervise les autres unités et effectue les traitements.
Mémoire : lieu de stockage des informations.

Entrée et Sortie : ce sont les unités qui sont destinées à recueillir les informations en entrée et à les restituer en sortie.
Les bus assurent les connections entre les différentes unités.

Configurations de base des architectures :

Les instructions à exécuter sont stockées dans les mémoires et les données sur lesquelles elles opèrent sont également stockées dans les mémoires. Instructions et données étant à priori 2 notions très différentes, il y a 2 possibilités :

-si on utilise 2 mémoires indépendantes l'une stockant le code exécutable et l'autre stockant les données, on a l'architecture HARVARD

-soit une seule et unique mémoire qui stocke à la fois le code et les données à des endroits différents, on a l'architecture de VON NEUMANN

Les configurations logicielles :

Architecture CISC (complex instruction set computer)

Ordinateur à jeu d'instructions complexes. Elle est utilisée par tous les processeurs de type x86 (INTEL , AMD , CYRIX ….). Les processeurs basés sur l'architecture CISC peuvent traiter des instructions complexes cad une instructions peut accomplir de nombreuses opérations , son jeu d'instruction est assez sophistiqué.

Architecture RISC (reduced instruction set computer)

Ordinateur à jeu d'instructions réduit. Ne possède que des instructions simples réalisant des opérations élémentaires et pas de fonctions supplémentaires câblées, cela impose des programmes ayant des instructions simples et cela se traduit par une programmation plus difficile et un compilateur plus puissant.

Architecture Non VON NEUMANN: Multiprocesseur et calcul parallèle

Une architecture multiprocesseur est un système informatique dans lequel plusieurs processeurs fonctionnent en parallèle. On trouve le système multiprocesseur avec mémoire commune partagée et le système multiprocesseur où chaque processeur possède sa propre mémoire locale et l'ensemble se partageant une mémoire commune.

Principe de base d'un architecte des ordinateurs :

L'architecte doit concilier les deux aspects : les besoins fonctionnels du programmeur et les contraintes financières.

Pourquoi étudier l'architecture des ordinateurs ?

- Etude et développement des compilateurs , des systèmes d'exploitation ……
- Bien comprendre les performances des systèmes informatiques
- Travailler avec le langage assembleur

SERIE N°1 DE TRAVAUX DIRIGES (2004/2005)

1. Définir « la structure des ordinateurs » et la situer par rapport à l'architecture des ordinateurs.
2. Enumérer des métiers ou spécialités correspondants aux différents niveaux de l'architecture en couches d'un système informatique.
3. Expliquer les termes « implémentation » et « réalisation » dans le cadre de l'architecture des ordinateurs .
4. Donner des exemples correspondants à la technologie des ordinateurs.
5. L'informatique est généralement associée avec la programmation. Cependant l'étude de l'architecture des ordinateurs est une partie importante de la formation des élèves informaticiens. Expliquer pourquoi c'est vrai
6. Quelles sont les techniques permettant de faire exécuter un programme écrit en langage L1 ?
7. En quel sens matériel et logiciel sont-ils équivalents ?
8. Expliquer les différences entre la machine de VON NEUMANN et la machine de HARVARD
9. Expliquer l'utilité du jeu d'instructions en architecture des ordinateurs
10. Soit un ordinateur dans lequel toutes les couches sont différentes (1 à 4). Chaque couche a des instructions M fois plus puissantes que celle de la couche immédiatement inférieure. Sachant qu'une instruction au niveau 1 prend K nanosecondes pour être exécutée Combien de temps prendrait l'exécution d'un programme de niveau 4 composé de N instructions ?
11. ***(Contrôle n°1 2002/2003)*** Soit le modèle simplifié en couches d'un système informatique :

5	Couche applications
4	Couche langages de haut niveau
3	Couche langage assembleur
2	Couche langage machine
1	Couche physique

Le niveau 5 possède des instructions 30 fois plus puissantes que le niveau 2
Le niveau 4 possède des instructions 12 fois plus puissantes que le niveau 2
Le niveau 3 possède des instructions 3 fois plus puissantes que le niveau 2

- Quels sont les moyens de passage entre un niveau et le niveau immédiatement inférieur ?
- Deux programmeurs Saber et Karim se sont engagés à réaliser un programme utilitaire. Saber a utilisé le niveau 4 et a conçu un programme de 400 instructions. Karim a utilisé le niveau 3 et a conçu un programme de 1000 instructions. Quel est le temps d'exécution des programmes de Saber et Karim ?
- Quelle est la puissance d'une instruction de niveau 4 par rapport à une instruction de niveau 3?
- Exprimer les gains de Karim en termes de rapidité d'exécution et de taille de programme ?

12. ***(Synthèse 2002/2003)*** Pourquoi utilise-t-on le modèle en couches pour la description d'un système informatique ? Si un programmeur est autorisé à programmer au niveau micro-architecture, quelles seront les caractéristiques des programmes réalisés?

13. ***(Rattrapage 2002/2003)*** Un système informatique peut être vu comme un empilement de couches ou de niveaux. Le langage du plus bas niveau est le plus simple, celui du plus haut niveau est le plus complexe du point de vue des concepteurs des ordinateurs :
    - A quel niveau du modèle en couches est localisée la machine réelle ?
    - Pourquoi les concepteurs d'ordinateurs choisissent un langage plus simple au bas niveau ? Est-ce que ce langage est pratique pour son utilisation par l'être humain ?
    - Comment l'ordinateur arrive t-il à exécuter des programmes écrits dans un langage de plus haut niveau ?

14. ***(Contrôle n°1 2003/2004)*** La compagnie A.M.S. a produit un prototype de machine selon le modèle simplifié suivant :

4	Langages haut niveau
3	Assembleur
2	Langage machine
1	Micro-architecture <micro-programmée>
0	Logique

Le niveau 0 a besoin de 25 ns pour exécuter une micro-instruction
Une instruction du langage machine a besoin de 12 micro-instructions
Un programme de Test a pris 30 s pour s'exécuter sur ce prototype

Le chef du projet non satisfait du résultat du test a réclamé plus d'efforts aux équipes. L'équipe n°1 travaillant au niveau 0 s'est engagée à diminuer de 5 ns le temps d'exécution de la micro- instruction et L'équipe n°2 travaillant au niveau 1 s'est engagée à réduire le nombre de micro-instructions à 10 micro-instructions pour chaque instruction du langage machine

1- Quel sera le temps d'exécution du programme de test si seulement l'équipe n°1 réussit ?
2- Quel sera le temps d'exécution du programme de test si seulement l'équipe n°2 réussit ?
3- Quel sera le temps d'exécution du programme de test si les 2 équipes réussissent ?
4- Le chef de projet a décidé de remplacer la micro-programmation avec une logique câblée et une instruction en langage machine nécessite alors 10 ns pour s'exécuter directement, quel sera le temps d'exécution du programme de Test ?
5- Quelle est la différence entre la micro-programmation et la logique câblée ?

T.P N°1 (2004/2005)

L'ORDINATEUR : POINT DE VUE DE L'UTILISATEUR

SARL TOTO REVENDEUR D'EQUIPEMENT INFORMATIQUE Cité des ordinateurs DDD TEL : 32.32.32.32
BENEFICIEZ DE SYSTEMES PERFORMANTS
MICRO ORDINATEUR TOTO 2003 COMPATIBLE IBM PC - BOITIER TOUR - PENTIUM III 667 MHZ - RAM 64 MB - DISQUE DUR 30 GB - CARTE GRAPHIQUE 3 D - CARTE SON + BAFFLES - LECTEUR DISQUETTES 3 ' ½ - LECTEUR CDROM 48 X - PORT PARALLELE , PORT SERIE , 2 PORTS USB - MONITEUR 17 '
PRIX : 3000,00 DA TTC

Q1 : Observez un PC puis décrivez les différents éléments qui le composent ?
Q2 : Décrivez les connexions entre les différentes parties
Q3 : Expliquez comment mettre en marche le PC et décrivez son éveil
Q4 : Expliquez comment faire travailler l'ordinateur
Q5 : Expliquez comment éteindre l'ordinateur
Q6 : Expliquez l'annonce du revendeur
Q7 : Expliquez les étapes à accomplir pour acheter un PC
Q8 : Citez 5 types d'utilisation possible d'un PC
Q9 : Citez des avantages de l'utilisation d'un PC
Q10 : Citez des inconvénients de l'utilisation d'un PC

MODELE EN COUCHES : MACHINE DES ENSEMBLES

Jeune architecte des ordinateurs débutant disposant d'un PC compatible IBM, des enseignants de la théorie des ensembles vous ont commandé une machine pour l'assistance de leurs élèves , cette machine doit disposer d'un langage propriétaire qu'ils ont défini comme suit :
C ; pour effacer l'écran de la machine
E1 = « entier » ; pour indiquer la taille du 1er ensemble
Puis la machine attend la saisie successive des éléments du 1er ensemble
E2 = « entier » ; pour indiquer la taille du 2eme ensemble
Puis la machine attend la saisie successive des éléments du 2eme ensemble
DIF ; pour calculer la différence symétrique des 2 ensembles et la machine affiche le résultat
INTER ; pour calculer l'intersection des 2 ensembles et la machine affiche le résultat
REU ; pour calculer la réunion des 2 ensembles et la machine affiche le résultat
F ; Fin de la session de travail
L'invite (prompt) de la machine est : **M-ENS>** ……

Ecrire un programme répondant aux besoins de ces spécialistes

Rappel :

La différence symétrique de 2 ensembles X et Y est l'ensemble des éléments de X et de Y qui ne sont pas communs à X et Y
L'intersection de 2 ensembles X et Y est l'ensemble des éléments communs à X et Y
La réunion de 2 ensembles X et Y est l'ensemble des éléments qui appartiennent à X ou à Y

ARCHITECTURE DE VON NEUMANN

HISTORIQUE

Les concepts de report automatique, d'exécution séquentielle et d'enregistrement de l'information et sa conservation pendant une certaine période, indispensables à la réalisation des ordinateurs étaient connus depuis fort longtemps.

DATES, INVENTIONS ET PERSONNALITES :

Dates, inventions et personnalités	Réalisations
Le boulier 3eme millénaire avant J.C	Réalise les opérations d'addition et de soustraction
Les Jaquemarts à partir du 14eme siècle	Automate sous forme de personnage :machine à programme interne figé qui exécute la même suite d'opérations tout le temps
Les Logarithmes (John Napier) 1614	John Napier trouve le moyen de réduire le travail qu'exige une longue multiplication ou une longue division en inventant les logarithmes : mise en œuvre de la règle à calcul
La Pascaline (PASCAL) 1642	1ere véritable machine mécanique capable d'additionner et de soustraire, de plus elle permet le report automatique
Machine à calculer (LEIBNIZ) 1673	Leibniz conçoit une machine pouvant effectuer les 4 opérations mathématiques
Métier à tisser (FALCON) 1728	Falcon utilise le carton perforé pour effectuer la commande automatique d'un métier à tisser
Métier à tisser (JACQUARD) 1801	Jacquard perfectionne et industrialise le système de Falcon, le procédé permet en plus de reproduire un tissu en plusieurs exemplaires
Machine analytique (BABBAGE) 1833	Babbage utilise le principe de la pascaline et les cartes perforées pour concevoir un appareil (machine analytique) qui réunissait les fonctions automatiques essentielles d'un ordinateur
L'algèbre de boole(GEORGE BOOLE)1854	GEORGE BOOLE publie un essai intitulé *Une étude des lois de la pensé*e, et présente une algèbre pour simuler les raisonnements logiques
Les appareils de HOLLERITH 1889	Hollerith utilise la carte perforée pour effectuer le dépouillement statistique d'un recensement de la population américaine. Création de 2 machines : une perforatrice et une trieuse, invention de la carte perforée à 80 colonnes
L'Ordinateur MARK I (AIKEN) 1944	Le professeur Howard Aiken de l'université HARVARD construit avec le concours d'IBM un calculateur électromécanique entièrement automatique. Le programme est communiqué à la machine sur des rubans perforés Addition 1/3 s , multiplication 4s , division 11s
L'Ordinateur I.A.S (VON NEUMANN) 1945	Description des composants essentiels d'un ordinateur (architecture de Von Neumann), introduction de nouveaux concepts : le programme enregistré et la rupture de séquence
Commercialisation des Ordinateurs 1952	Début de la production en série des ordinateurs et leurs commercialisations UNIVAC-1 (48 exemplaires)

CLASSIFICATION TECHNOLOGIQUE :

Suite aux progrès considérables des propriétés des circuits : Miniaturisation, fiabilité, complexité et vitesse et à l'évolution de l'exploitation des ordinateurs, on distingue 4 générations d'ordinateurs :

Génération	Caractéristiques
1ere génération : 1945-1955	Composants : relais, tubes à vides, résistances Machines consommatrices d'énergie, volumineuses, peu fiables Instructions en langage machine seulement (0 et 1) IBM 700 , UNIVAC 1 Calcul des tables pour la balistique, la bombe H, etc..
2eme Génération : 1955-1965	Composants : transistors et circuit imprimés Stockage sur bande magnétique, Ecran Apparition des systèmes d'exploitation Programmation en assembleur puis FORTRAN(1957) puis COBOL(1959) Calcul numérique répétitif et applications de gestion
3eme Génération : 1965-1980	Composants : circuits intégrés (puces) Faible consommation énergétique, encombrement réduit, plus fiable Multiprogrammation, temps partagé, temps réel, communication réseaux ALGOL, LISP , APL , BASIC , PASCAL Ordinateur central d'entreprise, spécialisé en gestion ou calcul
4eme Génération : 1980- ???	Composants : microprocesseur VLSI (very large scale integration) : augmentation du nombre de transistors LOI de MOORE : le nombre de transistors intégrés dans une puce doublera tous les 18 à 24 mois Parallélisme massif Augmentation de la puissance de calcul Généralisation de l'ordinateur personnel : traitement texte, tableur Traitement d'images et de son Internet : navigateur WEB …

CLASSIFICATION DE FLYNN :

La manière dont sont traitées les informations dans l'ordinateur

SISD : Single Instruction Single Data stream (une seule instruction, une seule donnée). (von neumann)
MISD : , *Multiple Instruction Single Data strea*m, (instructions multiples, une seule donnée)
SIMD : *Single Instruction Multiple Data stream* (une instruction, plusieurs données). (traitement tableau)
MIMD : *Multiple Instruction Data stream* (instruction multiples, données multiples) .(multiprocesseurs)

PERFORMANCES DES ORDINATEURS :

Les performances des ordinateurs se mesurent au nombre d'instructions qu'ils peuvent traiter en une seconde. Les unités les plus simples sont les MIPS (million d'instructions par seconde) et le MFLOPS (million d'instructions flottantes par secondes).
Pour comparer entre eux divers ordinateurs, il est généralement admis que la seule manière correcte est la mesure du temps d'exécution sur des programmes réels pour des entrées déterminées (Benchmarks spécifiés par des consortiums industriels ou des organisations scientifiques)

UNITES DE MESURE UTILISEES EN INFORMATIQUE :

CAPACITE : 1 K (kilo) = $10^3 = 2^{10}$, 1 M (méga) = $10^6 = 2^{20}$, 1 G (giga)= $10^9 = 2^{30}$, 1 T (tera) = $10^{12} = 2^{40}$, 1 P (peta) = $10^{15} = 2^{50}$

TEMPS : 1 ms(milliseconde)=10^{-3} , 1 us (micro)= 10^{-6} , 1 ns (nano)=10^{-9} , 1 ps (pico)= 10^{-12}

LES ORDINATEURS ACTUELS :

TYPE	EXEMPLE d'APPLICATION
Ordinateur enfoui	Montres, voitures, poupées
Ordinateur de jeux	Jeux vidéo
Ordinateur personnel (micro-ordinateur)	Les ordinateurs de bureau, les portables
Serveur	Serveurs de réseau
Mainframe (énorme capacité stockage = tétras octet)	Traitement par lot dans une banque
Superordinateur	Prévision météo, simulation

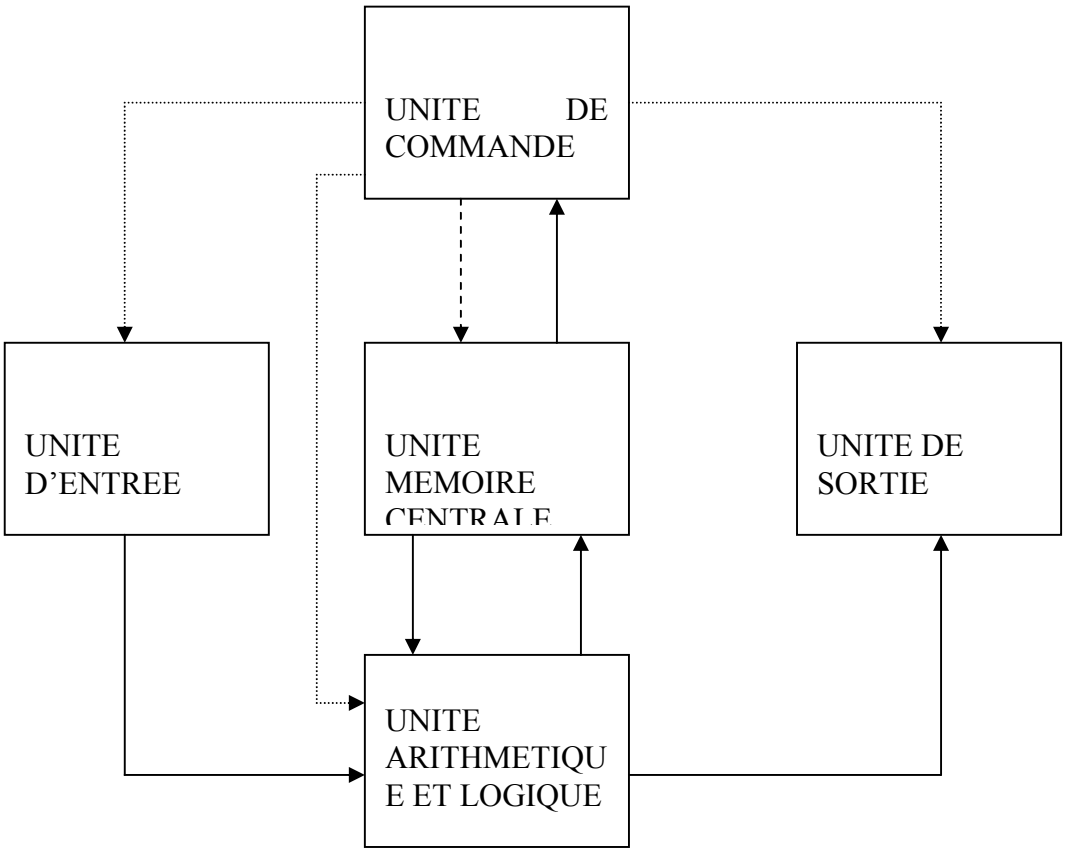
LE MODELE DE VON-NEUMANN :

Le modèle (architecture) de von neumann est composé de 5 composants essentiels :

- unité arithmétique et logique (UAL)
- unité de commande ou de contrôle
- unité de mémoire centrale ou mémoire principale
- unité d'entrée
- unité de sortie

Avec l'utilisation des concepts :

- programme enregistré en mémoire selon la numération binaire
- exécution séquentielle des instructions avec possibilité de rupture de séquence



Flot de contrôle

Flot Instructions/Données

UAL : réalise les opérations arithmétiques et logiques sous la direction de l'unité de contrôle

Unité mémoire : unité emmagasinant les instructions et les données

Unité Commande : coordonne les interactions entre toutes les autres unités

Unité Entrée/Sortie : permettant l'échange d'informations avec l'extérieur du système

Programme : Suite d'instructions effectuant un certain traitement

Instruction : Traitement effectué à un instant donné par le système (action atomique) (information traitante)

Données : souvent appelées opérandes (informations traitées) sur lesquelles la machine effectuera les traitements dictés par les instructions.

remarque : instructions et données sont stockées dans la mémoire => le programme peut modifier ses propres instructions

Ces dispositifs permettent la mise en œuvre des fonctions de base d'un ordinateur :

- Le stockage des informations
- Le traitement des informations
- Le mouvement des informations

Ainsi donc L'unité d'entrée pourvoie le système avec les instructions et les données qui sont ensuite enmagasinées dans l'unité mémoire , les instructions et données sont traitées par l'UAL sous la direction de l'unité de contrôle , ensuite les résultats sont émis vers l'unité de sortie

CYCLE d'EXECUTION DE VON NEUMANN :

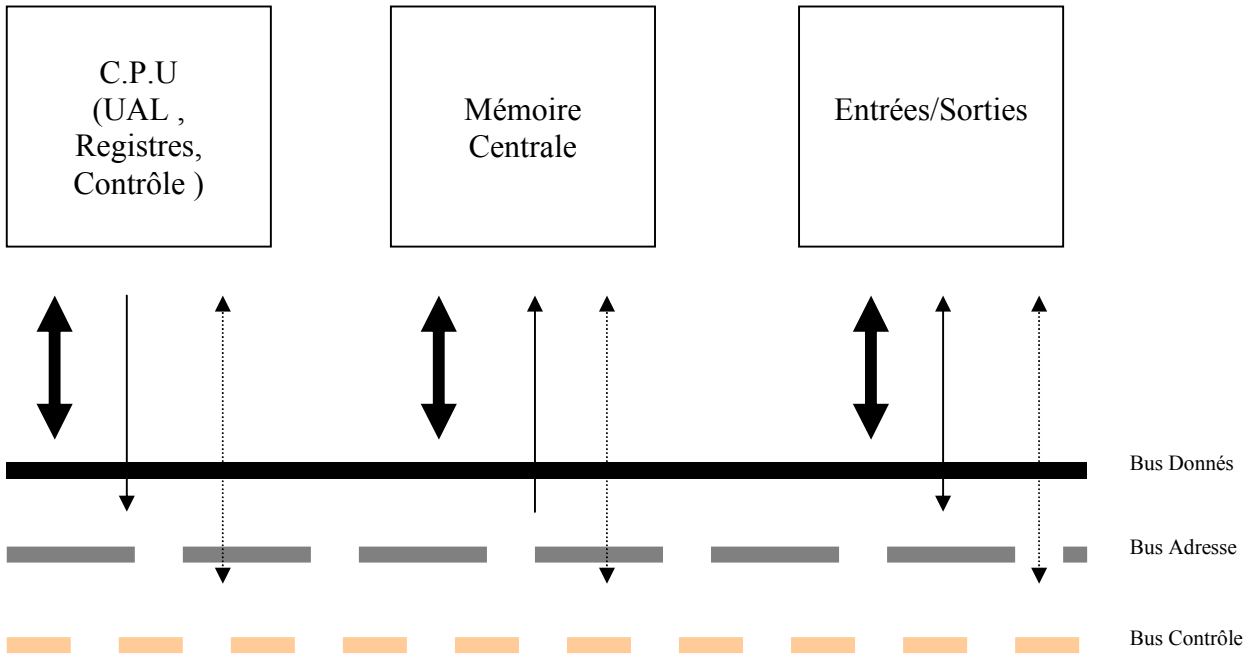
Instructions et données sont dans la mémoire centrale

Le fonctionnement schématique de l'ordinateur est le suivant :

1- Extraction de la prochaine instruction à exécuter de la mémoire
2- Analyse de l'instruction
3- Recherche dans la mémoire des données concernées par l'instruction s'il y a lieu
4- Exécution de l'opération et rangement du résultat
5- Retour à 1

LE MODELE A BUS SYSTEME :

Bien que le modèle de von neumann prévaut dans les ordinateurs modernes, il a été raffiné. Ainsi dans le modèle à bus système l'ordinateur a été divisé en 3 sous systèmes : Processeur (CPU) , Mémoire et les Entrées/Sorties .



Le point le plus important dans ce modèle est la structure d'interconnexion entre les unités qui se fait par un seul chemin partagé appelé bus système.

Physiquement un bus est un ensemble de fils électriques groupés par fonctions sur lesquels s'effectue la transmission des signaux en parallèle. chaque fil véhicule un bit

La caractéristique la plus importante du bus est sa largeur ou taille.

L'intérêt du bus système est de réduire le nombre d'interconnexions entre le CPU et les autres sous-systèmes

Le bus système permet 2 types de liaisons :

- des liaisons représentants des échanges d'informations

- des liaisons représentants des ordres

liaisons d'échanges d'informations :

- **Bus de données** : qui permet le transport des informations à transmettre !
- **Bus d'adresses** : qui identifie ou les informations sont à transmettre !

liaisons d'ordres :

- **Bus de contrôle** : dont la fonction est de contrôler l'accès et l'utilisation des bus d'adresses et de données et il décrit comment les informations sont transmises et de quelle manière

le bus de contrôle véhicule 2 types de signaux :

- des signaux de timing indiquant la validité des informations d'adresses et de données

- des signaux de commandes indiquant le type d'opération à effectuer

ex : ordre d'écriture de la donnée sur le bus donnée à l'adresse mémoire indiquée sur le bus adresse

remarque : tous les composants ou unités ne sont pas connectés au bus système de la même manière

ex : le CPU génère des adresses qui sont placés dans le bus adresse et la mémoire reçoit les adresses du bus adresse.la mémoire ne génère jamais d'adresses et le CPU ne reçoit jamais d'adresse c'est pour cela qu'il n'y a pas de connections correspondantes dans ces directions

Les types de transfert :

- Mémoire vers processeur : lecture d'instructions
- Processeur vers mémoire : écriture des données
- E/S vers processeur : lecture des données transmises par un périphérique
- Processeur vers E/S : transfert de données vers un périphérique
- E/S vers mémoire ou mémoire vers E/S : utilisation du principe d'accès directe à la mémoire pour faire communiquer les 2 unités sans passer par le processeur

UNITES D'ENTREES / SORTIES : (*Brève description le détail sera vu au chapitre 8*)

Pour pouvoir fonctionner correctement, l'ordinateur doit communiquer avec l'utilisateur. Cette communication est réalisée à l'aide des périphériques.

Les périphériques en entrée : permettent l'introduction des informations à la machine. Ces organes exécutent des ordres de lecture de l'information ex : clavier

Les périphériques en sortie : permettent la restitution des informations traitées sous une forma accessible à l'être humain. Ces organes exécutent des ordres d'écriture ex : imprimante

L'existence d'organes variés de communication avec la machine a 2 conséquences importantes :

1ere conséquence : les organes d'entrée /sortie ont des vitesses de fonctionnement très variables et souvent plus faibles que celle de l'unité centrale

2eme conséquence : l'information est représentée d'une manière différente selon les organes : le codage varie selon les supports et selon les organes

Un fonctionnement harmonieux du processeur impose qu'il reçoit l'information sous une forme unique avec le respect d'une vitesse homogène, pour cela on interpose entre les organes périphériques et le processeur des dispositifs particuliers appelés unité d'échange

L'unité d'échange est chargée de gérer les échanges d'informations avec le processeur, elle a pour rôle de décoder les informations en entrée pour les mettre en mémoire, ou encore de décoder les informations en sortie pour les envoyer aux périphériques.
L'unité d'échange joue le rôle de tampon pour absorber les différences de vitesse entre processeur et périphériques

LA MEMOIRE CENTRALE

C'est un organe qui est utilisée pour enregistrer, conserver puis restituer des informations (instructions et données)

La mémoire peut être vue comme un tableau linéaire de cellules (ou cases de rangement).chacune de ces cellules contenant une certaine quantité d'informations (ex 8 bits)

En principe la taille d'un emplacement mémoire pourrait être quelconque, mais en fait la plupart des ordinateurs en service aujourd'hui utilisent des emplacements mémoires d'un octet

Dans une mémoire de taille N emplacements, on a N emplacement mémoires, numérotés de 0 à N-1
Chaque emplacement est repéré par son N° appelé adresse

Le mot mémoire : est l'unité d'information accessible en une seule opération. sa taille varie d'une machine à une autre selon les possibilités du processeur

1 octet -> processeur 8 bits (Motorola 6502)
2 octets -> processeur 16 bits (Intel 8086)
4 octets -> processeur 32 bits etc…..

registre : est un élément mémoire de petite taille (1 ,2 , 4, 8 octets) et très rapide .

2 importants registres sont associés avec la mémoire :

- le **RAM** (registre d'adresse mémoire) contient l'adresse du mot à lire ou à écrire

- le **RIM** (registre d'information mémoire) reçoit l'information lue à partir de la mémoire ou destinée à y être écrite

Les caractéristiques de la mémoire centrale :

Capacité (taille) : elle représente le nombre d'informations stockable .exprimée en KO, MO

Temps d'accès : le temps nécessaire à une opération de lecture/écriture : le temps qui sépare l'instant ou l'opération est demandée (ex : adresse est présentée à la mémoire pour une opération de lecture) de l'instant ou l'opération est terminée (l'information est disponible)

Temps cycle mémoire : temps minimal s'écoulant entre 2 accès successifs = temps d'accès + temps de rafraîchissement mémoire

Vitesse de transfert (débit) : exprime la quantité d'informations lues/écrites par unité de temps

Opérations sur la mémoire centrale:

Chaque emplacement mémoire conserve les informations que le processeur y écrit jusqu'à coupure de l'alimentation électrique ou tout le contenu est perdu (principe de la volatilité)

Les seules opérations possibles sur la mémoire sont :

- Ecriture dans un emplacement : le processeur donne une valeur et une adresse et la mémoire range la valeur à l'emplacement indiqué par l'adresse
- Lecture d'un emplacement : le processeur demande à la mémoire la valeur contenue à l'emplacement dont il indique l'adresse. le contenu de l'emplacement lu reste inchangé

algorithme de lecture :

1- l'unité centrale commence par charger dans RAM l'adresse en mémoire du mot à lire
2- elle lance la commande de lecture à destination de la mémoire
3- l'information est au bout d'un délai égal au temps d'accès placée dans le RIM d'où l'unité centrale peut alors la récupérer

algorithme d'écriture :

1- l'unité centrale commence par placer dans le RAM l'adresse en mémoire du mot ou se fera l'écriture. elle fournit également l'information à stocker en la plaçant dans le RIM
2- il ne reste plus à l'unité centrale qu'à lancer une commande d'écriture qui aura pour effet de transférer l'information contenue dans le RIM vers le mot en mémoire centrale

remarque : dans l'étape N° 1, puisque les 2 opérations sont indépendantes et qu'elles utilisent des bus différents, elles peuvent être effectuées en parallèle -> gain de temps

Ordre de placement mémoire (big endian , little endian)

Comme il a été dit auparavant, la mémoire est en général organisée physiquement par octets càd que la plus petite unité adressable est l'octet bien qu'il existe des instructions qui manipulent des 16,32,64 bits
Comment représentés les informations représentées sur plus d'un octet et dans quel ordre ?

Ex : On veut placer la valeur 12345678 H à l'adresse X

Cette valeur se compose des 4 octets suivants 12,34,56,78
Ils seront placés en mémoire dans l'ensemble d'adresses { X , X+1 , X+2, X+3 }

2 solutions existent :

- Big endian : l'octet de poids fort est à l'adresse X ; on commence par le gros bout
- Little endian : l'octet de poids faible est à l'adresse X ; on commence par le petit bout

Notions de mémoire cache :

Le processeur est nettement plus rapide que la mémoire centrale. Si à chaque instruction, le processeur effectue un accès en lecture ou en écriture en mémoire centrale, il passerait la majeure partie de son temps à attendre les accès mémoires et on n'utiliserait pas pleinement ses possibilités.

Pour pallier à ce problème, l'idée est d'intercaler entre le processeur et la mémoire principale une mémoire plus rapide de petite taille. dans cette mémoire, on va essayer de garder les informations normalement en mémoire principale dont le processeur se sert le plus souvent à un instant donné.

Lors d'un accès par le processeur à un mot en mémoire 2 cas peuvent se rencontrer :

1- Le mot est présent dans le cache : le cache plus rapide envoie la donnée demandée sur le bus local et le CPU continue l'exécution
2- Le mot n'est pas présent dans le cache : l'accès mémoire se déroule normalement et l'information correspondante est alors délivrée simultanément au processeur et à la mémoire cache ou elle est stockée ainsi que les informations d'adresses voisines en mémoire centrale

Efficacité de la mémoire cache :

Règle 90/10 : en moyenne les programmes utilisent 90% de leur temps d'exécution pour accéder à 10% seulement du code

Principe de localité : on appelle localité d'un programme sa tendance à effectuer des accès mémoires répétées

Remarque : cette tendance augment le gain de vitesse apporté par le cache

Localité temporelle : il est probable qu'on accède encore à un mot déjà lu auparavant

Localité spatiale : il est probable si on a accédé à un certain mot, les mots les plus proches seront accédés

Remarque : l'architecture Harvard prouve son utilité dans l'utilisation du cache

LE CPU (unité centrale de traitement ou processeur) :

Au minimum, le CPU se compose d'une section de donnée qui contient l'UAL et des registres et une section contrôle qui interprète les instructions et gère le transfert entre registres.
La section de données est souvent appelée chemin de Données ou partie opérative

Les registres : ce sont des mémoires internes au CPU et sont utiles pour le stockage temporaire des informations pendant l'exécution d'une instruction

Le nombre et la taille des registres diffèrent d'un processeur à un autre

On distingue 2 types de registres :

- Registres visibles : leur nom (ou N°) peut être présent dans l'instruction. il est manipulable par le programmeur
- Registres non visibles : non visibles par le programmeur et servent à des besoins de stockage temporaire

UAL : peut être vue comme une fonction paramètres, on lui passe le type de l'opération, un ou plusieurs arguments et elle nous renvoie un résultat

Le choix du nombre d'opérandes est une question clé, il résulte d'un compromis entre le temps de calcul, le nombre d'opérateurs utilisés et le nombre de registres

Section contrôle : contrôle le mouvement des données et des instructions ainsi que les opérations de l'UAL et la synchronisation avec tous les composants de l'ordinateur

Bus interne : bus pour interconnecter les différents composants du CPU

Types d'instructions (*brève description le chapitre assembleur 8086 leur est réservé*)

Schématiquement, nous distinguerons trois grands types d'instructions :

1- les instructions de traitement portant sur des opérandes en mémoire comprenant essentiellement les opérations arithmétiques et logiques et les opérations de chargement rangement mémoire
2- les instructions de rupture de séquence permettant de rompre l'enchaînement séquentiel des instructions et de passer à une autre partie du programme si certaines conditions sont réalisées
3- les instructions d'échange permettant les échanges d'informations entre l'ordinateur et le milieu extérieur

Présentation simplifiée d'un processeur

(*le schéma sera présenté pendant la séance de cours*)

Le schéma représente l'architecture interne simplifiée d'un processeur à accumulateur.

Les informations circulent à l'intérieur du processeur sur un bus interne permettant l'échange de données entre les différentes parties du processeur

Les registres :

ACC : registre appelé accumulateur : c'est une des 2 entrées de l'UAL. il est impliqué dans presque toutes les opérations de l'UAL

RTUAL : registre tampon (temporaire) de l'UAL, stocke temporairement l'un des 2 opérandes

Registre d'état : constitué d'une suite de bits indépendants (des indicateurs ou drapeaux) positionnés par l'UAL à l'issue de chaque opération pour indiquer l'état du résultat et les conditions dans lesquels s'est terminée l'opération : résultat nul, négatif, débordement, retenue etc.……

RI : registre instruction, contient le code de l'instruction en cours d'exécution transféré à partir de la mémoire (*le codage des instructions sera vu dans le prochain chapitre*)

CO : compteur ordinal ou pointeur d'instruction : il contient l'adresse de l'emplacement mémoire ou se situe la prochaine instruction à exécuter. après chaque utilisation il est incrémenté si le programme est exécuté en séquence. il peut être modifié par les instructions qui réalisent des branchements.

Tampon : registre tampon (temporaire) d'adresse ou de donnée utilisé pour l'accès mémoire

L'UAL

l'unité arithmétique et logique est chargée de l'exécution des opérations booléennes et des opérations arithmétiques entières et flottantes. elle est composée de plusieurs opérateurs spécialisés chacun dans l'exécution d'un ou plusieurs types d'opérations

Ex : un additionneur est un circuit logique qui lorsqu'il reçoit à ses entrées les impulsions correspondantes aux nombres a et b , émet en sortie les impulsions correspondantes au nombre a+b

Remarque : les 1er ordinateurs disposaient d'opérateurs spécifiques câblés, aujourd'hui dans les ordinateurs modernes on trouve de moins en moins de circuits de ce type. Il n'existe plus de circuits propres à la multiplication ou à la division.
L'opérateur de multiplication est réalisé par addition-décalage
L'opérateur de division est réalisé par soustraction-décalage

Unité de commande :

Horloge : l'horloge (base de temps) divise le temps en battements de même durée appelés cycles. Elle distribue des impulsions régulièrement pour synchroniser les différentes opérations élémentaires à effectuer pendant le déroulement d'une instruction. A chaque cycle d'horloge, et en fonction des ordres données par l'unité de contrôle, certaines portes (principe des vannes) se ferment d'autres s'ouvrent pour laisser circuler des informations.

Cycle machine : cycle de base ou élémentaire égal à l'inverse de la fréquence

Cycle instruction : cycle fetch + cycle execute : chacun d'eux peut nécessiter plusieurs cycles machine

Cycle CPU : temps d'exécution de l'instruction la plus courte

Remarque : la vitesse de fonctionnement d'un ordinateur ne dépend pas seulement de sa fréquence d'horloge mais aussi du cycle mémoire et de la vitesse du bus

Décodeur : appelé décodeur de fonctions capable de reconnaître la fonction indiquée par l'instruction à exécuter parmi toutes les opérations possibles. la fonction de décodage consiste à faire correspondre à un code présent en entrée sur n lignes une seule sortie active parmi les N= 2 n sorties possibles

Séquenceur : le séquenceur est un automate distribuant des signaux de commandes aux diverses unités participantes à l'exécution d'une instruction selon un chronogramme précis en tenant compte des temps de réponse des circuits sollicités. il peut être câblé ou microprogrammé

Un séquenceur câblé est un circuit séquentiel complexe comprenant un sous circuit pour chacune des instructions à commander. ce sous circuit est activé par le décodeur

Il est toujours possible de remplacer un circuit logique par une suite de micros instructions stockées dans une mémoire de microprogrammation. Le code de l'opération à exécuter dans l'instruction est utilisé comme étant l'adresse de la 1ere micro instruction du microprogramme. Ce microprogramme est capable de générer une suite de signaux de commande équivalente à celle produite par un séquenceur câblé.
Un séquenceur microprogramme est plus lent qu'un séquenceur câblé.

Différentes phases de traitement d'une instruction

Le déroulement d'une instruction peut être décomposé en trois phases :

- phase de recherche et d'analyse de l'instruction
- phase de recherche de l'opérandes ou des opérandes et traitement effectif de l'instruction
- phase de préparation de l'instruction suivante

Chaque phase comporte un certain nombre d'opérations élémentaires exécutées dans un ordre précis par l'unité de commande.

Cas d'une instruction arithmétique à un seul opérande

Phase 1

1.1 (CO) ->RAM
1.2 commande de lecture
1.3 (RIM)->RI
1.4 décodage du code opération

Phase 2

2.1 adresse opérande->RAM
2.2 commande de lecture
2.3 (RIM)->UAL
2.4 exécution de l'instruction

Phase 3

3.1 (CO)+1->CO

rq : 3.1 peut être effectué en même temps que 1.2

SERIE N°2 DE TRAVAUX DIRIGES (2004/2005)

1 : Expliquer les termes suivants : multiprogrammation, batch, temps partagé, temps réel
2 : Représenter algorithmiquement un ordinateur
3 : Un des concepts de l'architecture de VON NEUMANN est la possibilité de modifications des instructions. Expliquer comment et pourquoi ?
4 : Plus le nombre d'équipement connecté à un bus est grand, plus les performances décroissent. Expliquer comment et proposer une solution pour remédier à ce problème
5 : En utilisant les concepts vus en cours, représenter schématiquement une machine SIMD,MIMD
6 : Expliquer la différence entre le modèle de VON NEUMANN et le modèle à bus système
7 : Combien de mots mémoire peuvent être adressés par un processeur doté de 16 lignes d'adresse
8 : Un processeur possède 8 lignes de données, combien d'opérations de lecture sont requises pour lire un mot de 16 bits ? Votre conclusion !
9 : Pour les 2 modèles (VON NEUMANN et à bus système) expliquer les étapes nécessaires pour lire 100 octets de données à partir d'un périphérique d'entré et les faire sortir après par un périphérique de sortie. on suppose qu'on a au moins 100 octets de mémoire disponible et toutes les entrées doivent être lues avant le commencement de toute opération de sortie
10 : Un ordinateur contient 1024 cellules de 8 bits chacune. au minimum combien de bits d'adresse doit posséder cet ordinateur
11 : La fréquence des impulsions d'une horloge est de 1000 Mhz .calculer le temps nécessaire pour le traitement d'une instruction à 5 cycles machines
12 : Que se passerait-il si l'étape « modification du compteur ordinal pour qu'il pointe sur l'instruction suivante était oubliée ?
13 : Si le registre d'adresse mémoire comporte 32 bits, calculer :

a- le nombre de mots adressables si 1 mot=1 byte , la taille de la mémoire
b- la plus haute adresse possible pour ces mots de 1 byte
c- le nombre de mots adressables si 1 mot=32 bits, la taille de la mémoire
d- la plus haute adresse possible pour ces mots de 32 bits

14 :Soit une machine dotée d'une mémoire centrale de 512 K mots de 32 bits , sachant que l'instruction type occupe un mot mémoire , quelles tailles proposeriez-vous pour les registres CO et RI
15 : Si la largeur d'un bus de donnée est de 8 bits et la longueur de toute instruction est de 16 bits. Expliquer la performance de ce système
16 : Donnez l'algorithme concernant une instruction de rupture de séquence
17 : Pourquoi on utilise généralement une UAL à 2 entrées au lieu de 3 ou plus ? expliquer pour le cas d'une addition
18 : Quelles sont les multiples possibilités offertes par la microprogrammation ?
19 : Quelle est la différence entre le bus système et le bus interne ?
20 : Est-ce que une mémoire cache est plus rapide qu'un registre ? Pourquoi ?
21 : En utilisant la notation vue en cours de transfert entre registres, décrire le déroulement d'une instruction de multiplication , addition et l'instruction ET logique
22 : Un petit ordinateur possède une mémoire de 16 mots de 8 bits.

a- Combien d'octets, de bits, de Kilo octets et de Méga octets contient cette mémoire?
b- Combien de valeurs différentes peut prendre un mot de cette mémoire ?

23: ***(Contrôle n°1 2002/2003)***
Soit un petit ordinateur à architecture bus système composé d'un processeur travaillant à une fréquence de 8 MHZ et d'une mémoire centrale de 2 mega mot mémoire d'un octet. Une instruction se compose de 16 bits.

1- Trouver la taille du RAM, RIM, CO, RI, bus d'adresses et bus de données
2- A quel niveau de l'ordinateur la différence est faite entre une instruction et une donnée ?

3- Si on suppose que la lecture d'une donnée de la mémoire vers le processeur nécessite 2 cycles machines, quel sera le temps de recherche d'une instruction ?
4- En réalité la taille d'une instruction est variable. Si la valeur contenue déjà dans CO est 15_H et la valeur significative contenue dans RI est 231415_H quel sera le contenu de CO pour la recherche de la prochaine instruction ? (Suggestion : dessiner le plan mémoire)
5- Définir le modèle du double cache (cache pour instructions et cache pour données). Quelle est l'utilité de ce modèle ?

24 : ***(Synthèse 2002/2003)***
Soit une machine à accumulateur disposant d'une mémoire de 4 MO avec des mots mémoires composés de 32 bits. Cet ordinateur possède 1024 instructions et 3 modes d'adressage . son registre d'instruction se compose de 3 champs (code opération , mode d'adressage , adresse)
(**1 M = 2^{20}**)

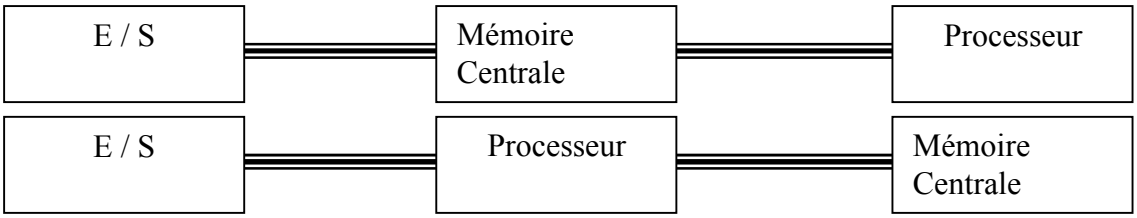
a- combien de mots cet ordinateur peut-il mémoriser ?
b- calculer la taille des registres CO , RIM , RAM , Accumulateur
c- détailler le format du registre d'instruction (taille de chaque champ)

25 : ***(Rattrapage 2002/2003)*** Soit une machine à accumulateur possédant les caractéristiques suivantes :

Une architecture à bus système
une fréquence d'horloge de 10 MHZ
le transfert d'une donnée entre la mémoire et le processeur nécessite un cycle machine
une mémoire centrale de 2 MO , la taille du mot mémoire est un octet
les modes d'adressage : immédiat, direct, indirect, auto-incrément, auto-décrément
256 types d'opérations
une instruction est formée des 3 champs suivants
(code opération, modes d'adressage, opérande immédiat ou adresse de l'opérande)

a) Quels sont les concepts introduits par VON NEUMANN en architecture des ordinateurs ?
b) Calculer la taille du CO, RIM , RAM , Accumulateur
c) Calculer la taille du registre d'instruction en précisant la taille de chaque champ
d) Quelle est valeur maximale de l'opérande dans le cas d'un adressage immédiat ?
e) Quel est le temps de recherche d'une instruction de la mémoire vers le processeur ?

26 : ***(Contrôle n°1 2003/2004)*** Les Questions sont indépendantes
1-Quel est le meilleur des 2 modèles suivants ? Pourquoi ?



2-Si le processeur a une fréquence de 10 MHZ et le cycle fetch + execute nécessite 5 cycles machines, combien de cycles machines aura besoin un programme de 10 instructions ?
3-Pourquoi utilise-t-on un registre d'état dans le processeur ?
4-Comparer la mémoire centrale, le registre et le cache en termes de capacité et de rapidité
5-On intercale entre la mémoire centrale et le processeur une mémoire cache, le temps d'accès au cache est égal à ¼ du temps d'accès à la mémoire centrale qui est égal à 10 ns. Si on suppose que le taux de succès du cache est 80%, quel sera le temps moyen d'accès mémoire ?

REPRESENTATIONS DES INFORMATIONS

Système de numération : Il est défini par un ensemble de chiffres et un ensemble de règles permettant d'écrire tout nombre à partir de ces chiffres.
Remarque : en base b on utilise b symboles

Règles
Soit b une base de numération et soit N un nombre à représenter il existe une suite unique d'éléments de l'ensemble des chiffres de ce système tel que :

$N = a_{n-1}\, b^{n-1} + a_{n-2}\, b^{n-2} + \ldots\ldots\ldots\ldots a_1\, b^1 + a_0$
$N = a_{n-1}a_{n-2} \ldots\ldots\ldots\ldots a_2a_1a_0$

Les a_i sont les symboles ou les éléments avec i qui représente l'ordre ; n représente le poids

Ex : $75362_{10} = 7.10^4 + 5.10^3 + 3.10^2 + 6.10^1 + 2.10^0$

Système de numération décimal

Base : 10
Symboles : 0,1,2 ……,9

Remarque : Inconvénient technologique pour son utilisation dans un système informatique
Exercice : Pourquoi ?

Système de numération binaire

Base : 2
Symboles : 0,1

Addition

+	0	1
0	0	1
1	1	10

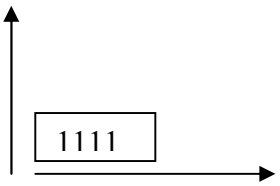
1 : retenue

Multiplication

X	0	1
0	0	0
1	0	1

Conversion décimal-binaire : On utilise des divisions entières successives par 2
remarque : Pour automatiser l'opération on continue jusqu'au quotient = 0 et on prend les restes
Ex : 15 10 = ?? 2

N	:	2	Q	R
15		2	7	1
7		2	3	1
3		2	1	1
1		2	0	1



Conversion binaire-décimal : Application directe de la formule
Ex : $1101_2 = 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 = 8+4+1=13$

Cas des nombres fractionnaires :

$N = a_{n-1}\, b^{n-1} + \ldots\ldots\ldots + a_0b^0 + a_{-1}b^{-1} + \ldots\ldots\ldots + a_{m-1}b^{m-1}$

Partie entiere | Partie fractionnaire

Remarque : Pour la partie entière même principe qu'avant
Pour la partie fractionnaire on utilise la multiplication par 2

Ex $0,65_{10} = ??_{2}$

Partie entière 0 : 0
Partie fractionnaire :

Nombre	*	2	Produit	Garde	Reste
0,65		2	1,3	1	0,3
0,3		2	0,6	0	0,6
0,6		2	1,2	1	0,2
0,2		2	0,4	0	0,4
0,4		2	0,8	0	0,8
0,8		2	1,6	1	0,6
0,6		2	1,2	1	0,2
……		..	……..	……	……

$0,65_{10} = 0,10\underline{1001}_{2}$

Ex : $1101,1001_{2} = ??_{10}$
Application directe de la formule
Partie entière : $1*2^{3}+1*2^{2} + 1*2^{0} = 13$
Partie fractionnaire : $0,1001 = 1*2^{-1} + 0*2^{-2} + 0*2^{-3} + 1*2^{-4} = 0,5625$

$1101,1001_{2} = 13,5625_{10}$

Système de numération octal : Ecriture condensée des suites binaires (3 bits)
Base: 8
Symboles : 0,1,2,…..,7

Décimal	Binaire	Octal
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8

Conversion binaire-octal : On fait un regroupement de 3 bits à partir de la droite et on remplace chaque regroupement par son équivalent octal
Ex : $1100010101_{2} = 001\ 100\ 010\ 101_{2} = 1425_{8}$

Conversion octal-binaire : Eclatement de chaque chiffre octal en son équivalent binaire sur 3 bits
Ex : $17_{8} = 001\ 111_{2} = 1111_{2}$

Conversion octal-décimal : Même principe que binaire-décimal : application directe de la formule avec comme base 8

Exercice : $345_{8} = ??_{10}$

Conversion décimal-octal : Même principe que décimal-binaire : divisions successives entières par 8

Exercice : $4986_{10} = ??_{8}$

Système de numération hexadécimal : Ecriture plus condensée des suites binaires (4 bits)
Base: 16
Symboles : 0,1,2,…..,9,A,B,C,D,E,F

Décimal	Binaire	hexadécimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Conversion binaire-hexadécimal : On fait un regroupement de 4 bits à partir de la droite et on remplace chaque regroupement par son équivalent hexadécimal
Ex : $1010111011100_{2} = 0001\ 0101\ 1101\ 1100_{2} = 15DC_{16}$

Conversion hexadécimal-binaire : Eclatement de chaque chiffre hexadécimal en son équivalent binaire sur 4 bits

Exercice : $23AB_{16} = ??_{2}$

Conversion hexadécimal-décimal : Application directe de la formule avec comme base 16

Exercice : $345_{16} = ??_{10}$

Conversion décimal-hexadécimal : Divisions successives entières par 16

Exercice : $4986_{10} = ??_{16}$

Remarques : Cas des parties fractionnaires des nombres fractionnaires

Conversion binaire-octal : On fait des regroupements à gauche par tranche de 3 bits
Conversion binaire-hexa : On fait des regroupements à gauche par tranche de 4 bits
Conversion octal-décimal : Même principe que binaire-décimal : multiplication par 8
Conversion hexa-décimal : Même principe que binaire-décimal : multiplication par 16

Exercice : $10110110111,11101_{2} = ??_{8} = ??_{16}$

$270,14_{8} = ??_{10}$
$DADA,C_{16} = ??_{10}$

Codage des nombres : La représentation (codage) des nombres est nécessaire afin de les stocker et manipuler par un ordinateur. le codage doit s'effectuer sur un **nombre de bits fixé**

Code BCD (binary coded decimal) : décimal codé en binaire
Principe : Chaque chiffre de la base décimale est codé individuellement en son équivalent binaire sur 4 bits.
4 bits ➔ 2 4 ➔ 16 configurations ➔ 6 configurations en plus ➔ tenir compte pour les opérations arithmétiques

on évite la conversion binaire et on fait les opérations sur les représentations binaires

Décimal	Binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Ex : décimal 129
Binaire 1000001
BCD 0001 0010 1001
000100101001

Le code BCD est un code pondéré 8421 : les 4 bits nécessaires pour coder un chiffre décimal reçoivent des poids selon leurs positions

Ex : 8 ➔ 1000 DCB = 1*8 +0*4+0*2+0*1=8

Remarque 1: L'addition binaire de 2 chiffres décimaux en DCB ne donne pas un chiffre décimal en code DCB dès lors que le résultat dépasse 9

Ex : 8+7 = 15 en DCB 1000+0111 = 1111 inexistant en code DCB

On lui ajoute 6 (0110) càd 1111+0110 = 0001 0101 = 15

Exercice : essayer avec 231+467 ; 378+349 ; 4572+631

Remarque 2 : Pour la soustraction on soustrait 6 au résultat si on a dû faire un emprunt pour effectuer la soustraction

Exercice : essayer avec 20-8

Remarque 3 : Le code BCD se trouve dans les machines orientées vers la gestion ou les opérations arithmétiques sont beaucoup moins nombreuses que les opérations d'entrées/sorties (ex comptabilité)

Codage des entiers naturels : Les entiers naturels sont codés sur un nombre d'octets fixé (un octet est un groupe de 8 bits).On rencontre habituellement des codages sur 1,2 ou 4 octets. Un codage sur n bits permet de représenter tous les nombres naturels compris entre 0 et 2n -1

Exercice : intervalle de représentation pour 1 octet, 2 octets et 4 octets ?

On représente le nombre en base 2 et on range les bits dans les cellules binaires correspondant à leur poids binaire de la droite vers la gauche.Si nécessaire on complète à gauche par des zéros (bits de poids forts)

Ex : 170 10 sur un octet

1	0	1	0	1	0	1	0

Problèmes :
1-Tous les nombres ne peuvent être représentés
2-Risque de débordement lors des opérations

Exercice : essayer l'opération 170+192 avec un codage sur 1 octet

Codage des entiers relatifs : coder le nombre entier et son signe

1ere méthode : signe et valeur absolue
le bit de poids fort est un bit de signe (0 ➔ + , 1 ➔ -)

Ex : +4 10 et –4 10 sur un octet

+4 s

0	0	0	0	0	1	0	0

-4 s

1	0	0	0	0	1	0	0

Remarque : sur n bits on peut représenter $-(2^{n-1}-1)$ à $+(2^{n-1}-1)$
Exercice : intervalle de représentation sur 4 bits, 1 octet et 2 octets ?
Problèmes : - 2 représentations possibles pour 0
- comment additionner 2 nombres de signe différent ?

2eme méthode : complément à 1
On calcule le complément à 1 en remplaçant pour les valeurs négatives chaque bit à 0 par 1 et vice-versa
le bit de poids fort est un bit de signe (0 ➔ + , 1 ➔ -)

Ex : +2 10 et –2 10
+2 10 = 00000010 2 -2 10 = 11111101 2
remarques :
1- l'intervalle des entiers que l'on peut représenter en complément à 1 est le même que pour la représentation signe +valeur absolue
2- la soustraction est obtenue par addition du complément à 1 et report de la retenue. **Ex** : 5-2 = 0101+1101=1 0010 donc 0010+1=0011
3- le complément à 1 possède 2 zéros . *Exercice* : lesquels ?

3eme méthode : complément à 2
le complément à 2 est obtenu en additionnant +1 à la valeur du complément à 1
le bit de poids fort est un bit de signe (0 ➔ + , 1 ➔ -)

Exercice : complément à 2 de –6 sur 4 bits

remarques :

1- l'intervalle des entiers que l'on peut représenter en complément à 2 est compris entre -2^{n-1} à $+2^{n-1}-1$. *Exercice* : pourquoi moins un nombre par rapport Com 1?
2- la soustraction est obtenue par addition du complément à 2 en ne tenant pas compte de la retenue. **Ex** : 5-2 = 0101+1110=1 0011 donc 0011

Codage des nombres réels :

Virgule fixe : Les ordinateurs n'ont pas de virgule au niveau de la machine .les nombres sont considérés comme des entiers , le soin de placer la virgule étant laissée au programmeur (virgule virtuelle). Ce dernier doit connaître et faire évoluer au cours des opérations la place de la virgule.
Remarque :
on ne peut pas représenter les plus grands nombres ni les plus petites fractions car on est limité par le nombre de bit

Virgule flottante : La représentation en virgule flottante consiste à représenter les nombres sous la forme $sM * b^E$
s : signe du nombre ; M : mantisse du nombre ; E : exposant nombre ; b : base (2,10,…)

Cette manière de coder les nombres est à rapprocher des notations exponentielles utilisées en maths **Ex** : $976\,000\,000\,000\,000 = 9{,}76 * 10^{14}$
$0{,}0000000000000976 = 9{,}76 * 10^{-14}$
Nous avons déplacer dynamiquement la virgule à la place appropriée et on a utilisé l'exposant pour garder la trace de la virgule.

Normalisation : normaliser c'est conserver le maximum de chiffres significatifs possible
Un nombre binaire normalisé doit être de la forme $1{,}m * 2^{\pm E}$
m=bbbb…..b avec b : 0 ou 1

Comment coder E et M ➔ problème de standardisation pourquoi ?
Pour obtenir une représentation unique des nombres pour toutes les machines et éviter ainsi des calculs différents ➔ possibilité d'échange de données entre machines.

Standard IEEE 754-1985

Simple précision sur 32 bits

		e+127
1bit	8bits	23bits
S	E	M

Double précision sur 64 bits

	e+1023	
1bit	11bits	52bits
S	E	M

S : signe de la mantisse = 0 pour positif 1 pour négatif
E : Codage de l'éxposant par excédent (cad E+127 ou E +1023)
simple précision $0 < \text{exposant} < 255$

double précision 0 < exposant < 2047
M : partie fractionnaire de la mantisse normalisée
Remarque : comme le 1er bit de la mantisse normalisée est toujours 1, on n'a pas besoin de l'écrire ce qui permet de gagner un bit (23+1 ou 52+1) ➔ on l'appelle bit caché

Valeur d'un flottant

simple précision $(-1)^S * 2^{(e-127)} * 1, M$
double précision $(-1)^S * 2^{(e-1023)} * 1, M$

Exercice **:** l'intérêt de double précision **?**

Conversion décimal -> virgule flottante simple précision

Ex : $8,625_{10}$

1. Transformation en base 2 : 1000,101
2. Normalisation de la mantisse : $1,000101 \times 2^3$
3. Le signe + sera codé 0
4. Calcul exposant + 127 : 3+127=130
5. Écriture de l'exposant en base 2 : 130 = 10000010
6. Écriture du nombre : 01000001 00001010 00000000 00000000
7. Notation hexadécimale : $41\ 0A\ 00\ 00_{IEEE}$

Conversion virgule flottante simple précision-> décimal.

Ex : $C4B7DC21_{IEEE}$

1° : Inversion de l'ordre des octets : C4B7DC21

2° : Écriture en binaire : 1100 0100 1011 0111 1101 1100 0010 0001

3° : Extraction du signe : 0 : -

4° : Recherche de l'exposant : 10001001 : 137 en décimal, donc E = 137 - 127 = 10

5° : Extraction de la mantisse : **1.**011 0111 1101 1100 0010 0001

6° : Calcul de la mantisse en binaire :
1.011 0111 1101 1100 0010 0001 x 2^{10} = 1011 0111 110.1 1100 0010 0001

7° : Calcul de la partie décimale : 0. 1110 0001 0000 1= 0.879028…

8° : Calcul de la partie entière : 101 1011 1110 = 1470

Le résultat est donc : $-1470,879028\ldots_{10}$

Exercice : convertir -532_{10} en virgule flottante double précision

Opérations arithmétiques :

Addition :

1. dénormalisation pour avoir les deux exposants à la même valeur
2. additionner les mantisses
3. normalisation du résultat

Multiplication :

1. additionner les exposants
2. multiplication des mantisses
3. normalisation du résultat

Addition de deux nombres en virgule flottante IEEE de simple précision

Soit A = 40E9999A et B = BE99999A en virgule flottante IEEE. Effectuer A + B.
On a pour A :

s	e + 127	f de la mantisse
0	10000001	11010011001100110011010

e + 127 = 129, donc e = 2, et la mantisse est 1,11010011001100110011010.
Donc A = $1,11010011001100110011010 \times 2^2$

On a pour B :

s	e + 127	f de la mantisse
1	01111101	00110011001100110011010

e + 127 = 125, donc e = -2, et la mantisse est -1,00110011001100110011010.
Donc B = $-1,00110011001100110011010 \times 2^2$
On met B au même exposant que A en déplaçant la virgule de 4 positions vers la gauche.
On obtient alors B = $-0,00010011001100110011001100110011010 \times 2^2$
On effectue A + B (ici il s'agit d'une soustraction puisque B est négatif) :

$$\begin{array}{rl} 1,1101\ 0011\ 0011\ 0011\ 0011\ 010 & \times 2^2 \\ -\ 0,0001\ 0011\ 0011\ 0011\ 0011\ 001\ 1010 & \times 2^2 \\ \hline 1,1100\ 0000\ 0000\ 0000\ 0000\ 000 & \times 2^2 \end{array}$$

On n'additionne pas les quatre derniers bits de B, puisqu'ils sont hors des 23 bits significatifs.
On replace le résultat dans le format IEEE:
e + 127 = 129 = 10000001

s	e + 127	f de la mantisse
0	10000001	11000000000000000000000

Qu'on peut abréger en notation hexadécimale: 40E00000.

Multiplication de deux nombres en virgule flottante IEEE de simple précision

Soit A = 40A00000 et B = C0E00000. Effectuer A × B.

On a pour A :

Donc A = 1,01 × 2^2.

On a pour B:

Donc B = -1,11 × 2^2.

On effectue le produit :

$$\begin{array}{rl} 1{,}01 & \times 2^2 \\ -1{,}11 & \times 2^2 \\ \hline 101 & \\ 101 & \\ 101 & \\ \hline -10{,}0011 & \times 2^4 \end{array}$$

On normalise le résultat pour le mettre sous la forme 1,f et on obtient :

A × B = -1,00011 × 2^5

On replace ce résultat dans le format IEEE :

exposant + 127 = 5 + 127 = 132 = 10000100

Ou, en notation abrégée: C20C0000.

Cas particuliers

-infini	M=0,E =1111…11 -> débordement supérieur
-0	M=0,E =0000…00 -> impossible d'écrire 0
-NaN(not a number)	M<>0,E=1111…11 ->expression indécidable
-Dénormalisé	M<>0,E =0000…00 ->le résultat est dénormalisé

Codage des caractères

Le codage des caractères est réalisé à l'aide de tables qu'on trouve dans la majorité des livres.

Le code ASCII [American Standard Code for Information Interchange]

C'est le système de codage quasi universel. C'est un code à **7** positions, le 8ième bit étant réservé au bit de parité, ce qui fait **2^7=128** caractères représentables.Ce code comprend :
des fonctions de commandes (transmissions de données, déplacement écran ou imprimante : abulations, Retour chariot, sonnette..) ,des symboles de ponctuations (!,",},...) ,quelques symboles usuels en informatique ($,@,*,..) ,les chiffres ,les majuscules,les minuscules
Certains caractères ont une signification spéciale :

CODE	SIGNIFICATION
NUL	Caractère Nul
BEL	Sonnette(bip)
BS	Back Space
HT	Tabulation Horizontale
LF	Line Feed(Interligne)

VT	Tabulation Verticale
FF	Form Feed(Saut de Page)
CR	Carriage Return(Retour chariot)
ESC	Escape
SP	Space(Espace)

Remarque : chiffres < Majuscules <Minuscules
C'est l'ordre qui sera respecté dans la plupart des utilitaires de tri. Dans beaucoup de langages de programmation, on peut écrire if a<b pour des caractères, c'est cet ordre qui sera utilisé pour évaluer l'expression.

Certains constructeurs, dont IBM suivis par tous les fabricants de compatibles, ont enrichi cette table ASCII en utilisant le 8ème caractère, ce qui double le nombre de caractères représentables (256).

Les caractères supplémentaires sont essentiellement :
les caractères accentués utilisés dans la langue française notamment.
un jeu de caractères utilisés dans quelques langues (˜,v,...)
quelques symboles mathématiques (∃ ,∫)
caractères semi-graphiques qui permettent de réaliser des petits dessins géométriques (cadres, soulignés, etc..)
Il existe un grand nombre de **jeux de caractères**

Le code EBCDIC [Extended Binary Coded Decimal Interchange Code]
C'est un code à 8 positions, on peut donc représenter $2^8 = 256$ caractères. Ce code est surtout présent sur les gros systèmes, notamment IBM.

UNICODE
Le standard UNICODE est un système de codage de caractères conçu pour l'échange et l'affichage de textes écrits dans les multiples langues du monde moderne. C'est un système de codage sur **16 bits** qui permet de coder plus de 30 000 caractères couvrant la plupart des langages du monde.
Les 128 premiers caractères Unicode correspondant aux caractères ASCII. Les 128 suivants correspondent au jeu de caractères LATIN 1 de l'ASCII. On trouve ensuite les caractères cyrilliques, arméniens, hébreux, arabes, sanscrit, bengali, etc…
Le remplacement de l'ASCII par Unicode ne se fera pas du jour au lendemain mais les systèmes d'exploitation modernes comme NT ou Netware savent déjà tirer partie de ce codage.

Les codes redondants

Les informations sont stockées sous une forme binaire dans un ordinateur. Il arrive que les informations se corrompent un peu sous l'effet du bruit électrique. L'intégrité des données est assurée par l'utilisation des codes détecteurs et correcteurs d'erreurs qui reposent sur l'utilisation d'algorithmes plus ou moins complexes.

Principe des codes redondants : ces codes portent sur un nombre de bits supérieurs à celui strictement nécessaire pour coder l'information.
Aux **m** bits de données on ajoute **k** bits de contrôle, ce sont les **n** = m+k qui vont être transmis ou stockés en mémoire.
Codes de détection des erreurs ----------→ codes auto-vérificateurs
Codes de détection et de correction d'erreurs -----→ codes auto-correcteurs

Le code auto vérificateur de contrôle de parité :
Aux m bits d'information on ajoute 1 bit de parité qui tous forment le groupe d'information.
La parité est dite paire si le nombre de 1 dans le groupe est pair en tenant compte du bit de parité, elle est dite impaire dans le cas contraire.

Parité paire : on fixe le bit de parité à 0 ou à 1 de manière à obtenir un nombre nul ou pair de 1 en tenant compte du bit de parité
Parité impaire : on fixe le bit de parité à 0 ou à 1 de manière à obtenir un nombre nul ou impair de 1 en tenant compte du bit de parité

Ex : soient les 2 informations à transmettre

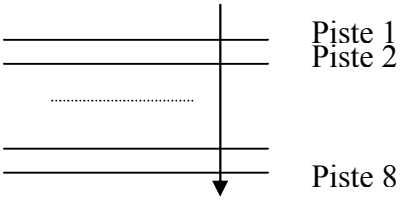
1er groupe : 1001011 (caractère K en ASCII) 2eme groupe : 0110111 (7 en ASCII)

parité paire : 1er groupe : **0**1001011 2eme groupe : **1**0110111
parité impaire : 1er groupe : **1**1001011 2eme groupe : **0**0110111

remarques :
1- quelle que soit la méthode utilisée (paire ou impaire) , le bit de parité fait partie intégrante de l'information à transmettre
2- si un bit est change par erreur → parité non vérifiée → erreur détectée
3- en cas d'erreur détectée → retransmission de l'information
4- le contrôle de parité ne permet de détecter qu'un nombre impair d'erreur

Exercice : expliquer la 4eme remarque

Le code auto correcteur de double parité : c'est le code obtenu en effectuant un double contrôle de parité (ligne et colonne)
le principe de la double parité est utilisé dans le stockage sur bande magnétique



- un caractère par colonne sur n-1 piste
- un bit de contrôle de parité sur la n^{ième} piste
- tous les m caractères on effectue un contrôle de parité

Ex : soit la représentation de 5 caractères (parité impaire)

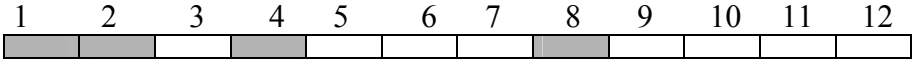
1	.....	0	0	0	0	0
1	…	0	1	1	1	1
1	…	0	1	1	1	1
1	…	0	1	0	1	1
0	…	0	0	1	0	0
0	…	0	0	1	0	0
1	…	0	0	0	1	1
0	…	1	0	1	1	0

Le bit N° 4 du 1er caractère est faux , on peut le corriger en 0

Code auto correcteur de Hamming :
Aux m bits d'informations on ajoute k bits de contrôle de parité . on a donc m+k = n bits à transmettre avec le respect de la règle que $2^k \geq n+1$ avec choix si possible de $2^k=n+1$
On numérote les bits de gauche à droite à partir de 1,les bits de contrôle (parité) sont placés sur les puissances de 2 (1,2,4,8,16 …..) , tous les autres bits sont des bits de données .

Chaque bit de parité contrôle un certain nombre de bits de données qu'on les déduit en construisant un tableau comme suit : en tête de tableau les positions des bits de contrôle et en ligne la séquence n des bits d'information en binaire

Ex : pour m = 8 bits on a k = 4 et n = 12 donc les bits de parité sont aux positions 1,2,4,8



Construisons le tableau pour connaître les positions contrôlées par les bits de parité

	8	4	2	1
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0

Chaque bit en tête de colonne contrôle les bits qui ont un '1' à la position correspondante

Bit 1 contrôle les bits 1,3,5,7,9,11
Bit 2 contrôle les bits 2,3,6,7,10,11
Bit 4 contrôle les bits 4,5,6,7,12
Bit 8 contrôle les bits 8,9,10,11,12

Parité paire : il faut mettre le bit de parité à 1 si le nombre de 1 dans les positions contrôlées est impair sinon il faut mettre 0 dans le cas si c'est pair

Ex : soit l'octet de données à transmettre : 10011010

Pour m=8 on a k=4 donc le mot (données et bits de parité) à transmettre est comme suit en laissant l'espace pour les bits de parité _ _ 1 _ 0 0 1 _ 1 0 1 0
Calculons la parité pour chaque bit de parité (? Représente la position à étudier)
la position 1 controle les bits 1,3,5,7,9,11:
? _ **1** _ **0** 0 **1** _ **1** 0 **1** 0. nbre de 1 = 4 mettre position 1 à 0: **0** _ **1** _ **0** 0 **1** _ **1** 0 **1** 0
la position 2 controle les bits 2,3,6,7,10,11:
0 **?** **1** _ 0 **0** **1** _ 1 **0** **1** 0. nbre de 1= 3 mettre position 2 à 1: 0 **1** **1** _ 0 **0** **1** _ 1 **0** **1** 0
la position 4 controle les bits 4,5,6,7,12:
0 1 1 **?** **0** **0** **1** _ 1 0 1 **0**. nbre de 1 = 1 mettre position 4 à 1: 0 1 1 **1** **0** **0** **1** _ 1 0 1 **0**
la position 8 controle les bits 8,9,10,11,12:
0 1 1 1 0 0 1 **?** **1** **0** **1** **0**. nbre de 1= 2 mettre position 8 à 0: 0 1 1 1 0 0 1 **0** **1** **0** **1** **0**
donc le mot complet à transmettre est : 011100101010

Correction d'erreur : on utilise l'algorithme suivant :

- pour le mot reçu, calculer tous les bits de parité
- si la parité est incorrecte , additionner toutes les positions ou le bit de parité est incorrect .le résultat obtenu nous donne la position du bit incorrect et on le corrige

Ex : si nous avons obtenu 011100101110 de l'exemple précédent
Position 1 : ok Position 2 : Non oK Position 4 : oK Position 8 : Non oK
Les positions 2 et 8 incorrectes donc 8+2= 10 donc le bit 10 est erroné on le corrige à 0

Exercice l'information suivante a été reçue: 000100111001101110 utilisant le code de hamming . Quelles sont vos conclusions ? Dans le cas où il y aurait une erreur, dire quelles sont les données que l'on aurait dû recevoir ?

Série n°3 de Travaux Dirigés (2004/2005)

1. Donner la valeur décimale des entiers suivants, la base dans laquelle ces entiers sont codés étant précisée.
(a) 1011011 et 101010 en binaire (base 2) ;
(b) A1BE et C4F3 en hexadécimal (base 16) ;
(c) 77210 et 31337 en octal (base 8).
2. Coder l'entier 2 397 successivement en base 2, 8 et 16.
3. Donner la valeur décimale du nombre 10101, dans le cas où il est codé en base 2, 8 ou 16. Même question avec le nombre 6535 codé en base 8 ou 16.
4. Combien d'entiers positifs peut-on coder en binaire sur un octet ? Combien de bits faut-il pour représenter 65 563 entiers différents en binaire ?
5. Coder en binaire sur un octet les entiers 105 et 21 puis effectuer l'addition binaire des entiers ainsi codés. Vérifier que le résultat sur un octet est correct. Même question avec les entiers 184 et 72.
6. Coder en binaire sur un octet les entiers 79 et 52 puis effectuer la multiplication binaire des entiers ainsi codés. Même question avec les entiers 135 et 46.
7. Indiquer la valeur codée par le mot de 16 bits 1101100101110101 suivant qu'il représente un entier non signé, ou un entier signé. Même question avec le mot 1001000011101101.
8. Indiquer la valeur codée par la suite 1101100101110101 qui représente un entier signé en complément à 2 sur 16 bits. Même question avec la suite 1001000011101101.
9. Représentation binaire des entiers négatifs
(a) Coder sur 4 bits les entiers 7, 2, 0, - 2, - 7 et - 8 avec les représentations suivantes :
– signe et valeur absolue ; – complément à 1 ; – complément à 2.
(b) Coder les entiers 61 et - 61 sur un octet en utilisant la représentation par le signe et la valeur absolue. Montrer que l'addition binaire de ces entiers ainsi codés produit un résultat incorrect. Montrer qu'en revanche le résultat est correct si ces entiers sont codés en utilisant la représentation par le complément à 2.
10. Effectuer en binaire (8 bits) les opérations 1 - 2, 51+127, - 3 - 127, -127+127, - 63 - 63. Préciser, pour chaque opération, la retenue et le débordement.
11. Représentation des réels
(a) En virgule fixe, décoder le nombre binaire 11.011 puis coder en binaire le réel 11.625.
(b) En virgule flottante normalisée, coder en binaire au format simple précision le réel 12.575 puis effectuer le codage inverse.
12. Opérations en virgule flottante.
Soit a =o.10010 x 10^{101} et b = o.11010 x 10^{1} Calculer $a+b$ et $a*$ b.
13.Coder pi selon la norme IEEE (pi=3.125) . Donner le résultat sous forme Hexadécimal
14. calculer 25 H + 5A H et B5 H + 4A H en utilisant un mot de 8 bits
15. Etant donnée la représentation signée avec point fixe en base 10 avec 3 bits à gauche et à droite du point décimal
(a) calculer la plus grande valeur positive et la plus petite valeur négatif (calcul des limites)
(b) calculer la différence entre 2 nombres adjacents (calcul de la précision)
16.soit l'exemple vu en cours de l'addition de 2 flottants, convertir le résultat obtenu au système décimal puis effectuer l'addition des 2 nombres dans le système décimal
Comparer les 2 résultats obtenus, Quelle est votre conclusion ?
17. En virgule flottante double précision Quel est le plus petit nombre normalisé ? le plus grand ?
18. Quelle est la valeur décimale du nombre en virgule flottante simple précision IEEE 754 suivant : 80000008 $_{IEEE}$
19.Dans la norme IEEE 754 pourquoi on a choisi l'ordre de représentation suivant
Signe exposant excédentaire mantisse au lieu de signe mantisse exposant excédentaire ?
20.Pourquoi le code BCD n'est pas un vrai code binaire ?
21.Pourquoi l'exposant est-il biaisé en virgule flottante (utilisation de l'excédent) ?
22.En virgule flottante simple précision norme IEEE 754
Quel est le plus petit nombre codable ?
Est ce qu'on peut coder 10^{-45} ?
Que représente 3,4 x 1038 ?
Est ce qu'on peut coder les nombres inférieurs à – 3,4 x 1038 ?

23.Quand est ce qu'on détecte un code impossible avec la méthode de Hamming ?
24.Soit la suite des bits de données à transmettre suivante : 1111000010101110
coder le message en utilisant le code auto-vérificateur de contrôle de parité paire et impaire et le code de Hamming de parité paire
25.Combien d'erreurs peuvent être détectées par un contrôle simple de parité sur un seul message (sans localisation) ? Est-il possible de corriger les erreurs ?
26. Supposons qu'on a reçu le code suivant utilisant la méthode de Hamming :
10010111000001101110 1
Détecter si c'est correct ou non.si c'est incorrect quel est le bit en question et le corriger.
27. ***(Contrôle n°1 2002/2003)***
A : soit la suite des 8 bits suivante : 10101110

Indiquer la valeur codée par cette suite si elle représente :

- entier avec signe et valeur absolue - entier en complément à 1 - entier en complément à 2
- un nombre BCD - un message reçu utilisant le code de hamming

B : Pour simplifier l'étude de la norme IEEE pour la représentation des nombres flottants, on utilise un petit nombre de bits par rapport au standard réel 754 selon le format suivant :

1 bit signe	3 bits exposant excédent 3	4 bits partie fractionnaire de la mantisse

L'exposant est représenté en excédent 3 et la représentation de la mantisse utilise le bit caché.

Par exemple 01011000 est la représentation du nombre + 6 dans ce format

l'arrondi est réalisé par simple troncature (çàd on ignore les bits à droite qui dépassent la capacité de représentation de la partie fractionnaire)

1- Donner la représentation de + 9,5 et - 1 dans ce format
2- Calculer
   - s1 = 1,1111 2 X 2 0 + 1,0000 2 X 2 – 4
   - s2 = s1 + 1,0010 2 X 2 4
   - s3 = 1,1111 2 X 2 0 + 1,0010 2 X 2 4
   - s4 = s3 + 1,0000 2 X 2 –4
3- Comparer s2 et s4. Est ce la représentation flottante possède la propriété d'associativité dans l'addition ? Pourquoi ?

Rappel : associativité (a+b)+c = a +(b+c)

LANGAGE MACHINE ET MODES D'ADRESSAGE

Chapitres 4 et 6 (10%)

▲ *les schéma seront dessinés pendant la séance de cours*

Introduction et Définitions :

Un processeur est capable d'effectuer un certain nombre d'opérations. Ces opérations sont déterminées par les instructions qu'il est capable d'exécuter.

Une instruction est le traitement effectué à un instant donné et correspond à une action atomique.

L'ensemble des instructions du processeur forme un langage spécifique appelé jeu d'instructions.

Le jeu d'instructions est également appelé langage machine.

Langage machine : est le langage binaire avec lequel les instructions du CPU sont définies et stockées en mémoire.

Ex : 00000111 : peut correspondre à une opération d'addition lors de son interprétation au niveau du CPU

Chaque CPU possède son propre jeu d'instructions

Le jeu d'instructions correspond au point de vue du programmeur de la machine en faisant abstraction du cheminement interne des informations, il doit inclure :

- Quels sont les opérations à réaliser ?
- Le format des instructions (càd comment sont spécifié les instructions ?)
- Où résident les données et les résultats ?
- Comment accéder aux données ?

l'objet de ce cours est de répondre à toutes ces questions

Eléments d'une instruction machine :

Instruction machine : une instruction machine est une suite de bits, elle doit fournir au CPU toutes les informations nécessaires pour déclencher une opération élémentaire.

Les éléments d'une instruction machine sont :

- Code opération : spécifie l'opération à réaliser
- Référence de l'opérande source : une opération peut nécessiter un ou plusieurs opérandes, les opérandes sources représentent les entrées de l'opération
- Référence de l'opérande résultat : une opération peut produire un résultat
- Référence de l'instruction suivante : indique au CPU où chercher l'instruction suivante après l'exécution complète de l'instruction en cours

Représentation de l'instruction :

une instruction comporte un groupe de bits divisé en sous-groupes appelés champs
- le 1er champ représente le code opération
- les autres champs peuvent comporter des données ou l'identification des opérandes

la longueur du code opération détermine le nombre maximum d'opérations
pour n bits on peut avoir 2^n opérations

Opérande explicite : l'instruction spécifie la localisation de l'opérande

Opérande implicite : automatiquement l'instruction utilise une localisation de l'opérande connue d'avance ex : utilisation de l'accumulateur, d'un registre ..

le nombre des opérandes explicites et leurs localisation (mémoire ou registre) est un facteur très important dans la conception du jeu d'instruction du processeur .

remarque : pour éviter d'avoir affaire au langage machine difficilement manipulable par l'homme on utilise un langage symbolique équivalent appelé langage assembleur .

langage assembleur : représentation symbolique du langage machine d'une machine réelle
langage symbolique : représentation symbolique du langage machine d'une machine fictive

en utilisant cette remarque , les codes opérations sont représentés par des abréviations appelés mnémoniques indiquant les opérations

Ex :

- ADD	pour une opération d'addition
- SUB	pour une opération de soustraction
- MUL	pour une opération de multiplication
- DIV	pour une opération de division
- LOAD ou LD	charger une donnée de la mémoire
- STORE ou ST	stocker une donnée dans la mémoire
- JUMP	branchement à une adresse donnée

les opérandes sont aussi représentés symboliquement

Ex : ADD R , y ! additionner la valeur contenue dans la position y au contenu du registre R

remarque : la définition d'un langage symbolique pour une machine fictive doit être explicite càd adjoindre à tout code symbolique le véritable sens du code opération et des opérandes

Classification des machines par le nombre d'opérandes :

il est parfois fait référence à une machine par le nombre de champs opérandes contenus dans ses instructions . on parle ainsi de machines à 4 , 3 , 2 ou 1 adresse (s) .

machines à 4 adresses :

A1 : adresse du 1er opérande ou la donnée elle même
A2 : adresse du 2eme opérande ou la donnée elle même
A3 : adresse où doit être rangé le résultat
A4 : adresse de l'instruction suivante

Effet : A3 <-- (A1) + (A2)
Inst Suiv = A4

Ex : ADD 19,13,100,110

Machines à 3 adresses :

Le champ A4 n'existe pas . l'adresse de l'instruction suivante étant indiquée par le compteur ordinal (CO)

Effet : A3 <-- (A1) + (A2) CO <-- CO + 1

Ex : ADD 19,13,100

Machines à 2 adresses :

Les champs A4 et A3 n'existent pas . le résultat est rangé dans le mot dont l'adresse est contenue dans le champ A2

Effet : A2 <-- (A1) + (A2) CO <-- CO + 1

Ex : ADD 19,13

Machines à 1 adresse :

A1 : désigne l'emplacement du 1er opérande
Le 2eme opérande se trouve dans un registre , l'opération terminée , le résultat est rangé dans ce registre .
Sur certaines machines ce registre est appelé accumulateur

Effet : ACC <-- ACC + (A1)

Ex : ADD 19

Classification des machines selon le modèle d'exécution :

Les opérandes peuvent être situés soit dans un registre interne à l'unité centrale, soit dans la mémoire

Selon la façon dont l'UAL obtient ses opérandes, on distingue 4 types de machines :

1. Les machines à accumulateur : Ces machines ont un seul registre nommé accumulateur dans lequel le résultat d'un calcul peut être stocké. Toute opération se fait alors entre une case

mémoire et l'accumulateur, la valeur résultat étant remise dans l'accumulateur. D'autres instructions permettent de transférer le contenu de l'accumulateur.

2. Les machines à pile : (appelée machine à zéro adresse) : ces machines n'ont pas de vrais registres, toutes les opérations effectuent un transfert entre la mémoire et une pile, ou bien effectuent un calcul en prenant leurs opérandes sur la pile et en remettant le résultat sur la pile
3. Les machines à registres généraux : Ce sont des machines qui disposent de plusieurs registres qui peuvent tous jouer le rôle d'accumulateur. Différents choix sont possibles : en particulier le résultat d'un calcul peut être stocké dans un registre ou directement en mémoire. on distingue ainsi selon l'emplacement des opérandes et du résultat les machines de type mémoire – mémoire et les machines de type mémoire – registre
4. Les machines à chargement / rangement (LOAD / STORE) : La différence par rapport aux machines à registres généraux est qu'ici les seules instructions qui accèdent à la mémoire sont des transferts directs entre la mémoire et un registre sans calcul. Toutes les opérations de calcul se font entre registres et retournent le résultat dans un registre

remarque : les processeurs performants sont tous à architecture chargement / rangement excepté ceux de la famille INTEL qui sont à mi-chemin entre une structure à accumulateur et une machine à registres généraux

Les modes d'adressage des opérandes :

Par mode d'adressage, on désigne le chemin que doit emprunter l'unité centrale pour accéder à l'opérande. Il désigne comment le champ adresse de l'instruction est utilisé pour déterminé l'opérande

Remarque : le mode d'adressage se trouve indiqué au sein de l'instruction dans le code opération ou dans un champ sépare réserve à cet effet appelé conditions d'adressage

Adresse effective : correspond à l'adresse finale envoyée dans le RAM après des transformations du contenu de la partie adresse de l'instruction

Adressage Immédiat : il ne s'agit pas d'un adressage à proprement parler , la valeur de l'opérande étant contenue dans un champ de l'instruction réserve à l'opérande

Au lieu de designer un emplacement (registre , mémoire) où se trouve une valeur , on désigne la valeur elle même

Ex : LOAD # nbr ! ACC <-- nbr

Adressage direct : l'adresse de l'opérande est donnée dans 'instruction sous forme d'adresse mémoire

Ex : LOAD adr ! ACC <-- M[adr]
Adresse effective : adr

Adressage indirect : l'emplacement où se trouve l'adresse est désignée et non plus l'adresse elle même

Ex : LOAD [adr] ou LOAD @adr ! ACC <-- M[M[adr]]

Adresse effective : M[adr]

Adressage registre : l'adresse de l'opérande est donnée dans l'instruction sous forme d'identifiant ou n° de registre

Ex : LOAD R1 ! ACC <-- R1
Adresse effective : R1

Adressage registre indirect : l'emplacement où se trouve l'adresse est désigné et non plus l'adresse elle même

Ex : LOAD (R1) ! ACC <-- M[R1] si R1 contient une adresse mémoire
! ACC <-- R1 si R1 contient n° registre
adresse effective : (R1)

Adressage relatif : l'adresse réelle est l'adresse indiquée dans l'instruction ajoutée à une adresse de référence contenue dans un registre le plus souvent ou dans une case mémoire particulière

adresse effective = (base) + déplacement

Ex : LOAD 100 (R1) ! ACC <-- M[100 + R1]

Adressage indexée : on obtient l'adresse effective en ajoutant à la partie adresse de l'instruction le contenu d'un registre appelé registre d'index

Le registre d'index est passible d'un certain nombre d'opérations telles que chargement , lecture , incrémentation ou décrémentation de 1 et comparaison

Ex : LOAD (adr + R1) ! ACC <-- M[adr + R1]

Adressages auto-incrémenté et auto-décrémenté : pour parcourir une série de positions mémoire successives (ex accès à un tableau), il est utile d'incrémenter (décrémenter) une adresse avant ou après chaque accès . il s'agit de Adressage auto-incrémenté (auto-décrémenté)

Ex : ADD R1 , (R2) + ! R1 <-- R1 + M [R2]
! R2 <-- R2 + d d : facteur de déplacement

Longueur des instructions :

la longueur des instructions varie selon les processeurs.
Pour un même processeur cette longueur peut être fixe ou variable
Processeur INTEL (8086) : 1 à 4 octets
Processeur MIPS : 4 octets
En cas de longueur variable , l'information contenue dans le code opération doit être suffisante pour déterminer la longueur de l'instruction . la longueur est y implicitement indiquée

Caractéristiques des tailles fixes et variables :

Taille fixe :

- décodage facile
- programme plus long

- temps de chargement d'une instruction constant

taille variable :
- décodage compliqué
- programme de taille optimale

Machines RISC et machines CISC : on trouve 2 catégories de jeu d'instructions

CISC : machines microprogrammées possédants un jeu d'instructions sophistiquées permettant de nombreuses opérations avec une seule instruction et des modes d'adressages extrêmement complexes
Ces principales propriétés sont :

- l'accès mémoire est directement disponible pour la majorité des types d'instructions
- les modes d'adressage substantiel en nombre
- les formats des instructions sont différents en longueur
- les instructions réalisent les opérations élémentaires et les opérations complexes

RISC : machines câblés motivées par la réduction du temps de conception des machines et la simplification de l'écriture des compilateurs. Ces principales propriétés sont :

- les adresses mémoires sont restreintes à LOAD / STORE et la manipulation des données sont des instructions registre à registre
- les modes d'adressage sont limités en nombre
- les formats des instructions ont tous la même longueur
- les instructions réalisent les opérations élémentaires
- nombre élevé de registres tous équivalents

Série n°4 de Travaux Dirigés (2004/2005)

Exercice 1. Machine fictive

La machine sur laquelle porte l'exercice comporte les éléments suivants :

- une mémoire principale de 32 octets
- un registre de travail de 8 bits (nommé ACCU) qui contient une des entrées de l'UAL avant exécution de l'opération et le résultat de l'opération ensuite
- une Unité Arithmétique et Logique (UAL) sachant exécuter les opérations suivantes : Addition, Soustraction
- un registre de 2 bits (nommé FLAGS) qui sont positionnés en fonction du résultat de l'UAL (Bit0=1 si le résultat est nul, Bit1=1 si le résultat est négatif)
- un registre compteur d'instruction (IP) qui contient l'adresse de la prochaine instruction à exécuter
- un registre instruction (INST) qui contient le code de l'instruction courante.

1. Combien de bits faut-il pour représenter une adresse ?
2. En supposant que le nombre d'instructions disponible est de 8 et que chaque instruction est représentée sous la forme Codop Adresse, combien de bits faut-il pour coder une instruction ?

La liste des instructions est la suivante : addition, soustraction, chargement du registre ACCU, rangement de ACCU, branchement, branchement si nul, branchement si négatif, fin.

1. Donner un code binaire (Codop) et un nom symbolique à chacune d'entre elles.
2. Ecrire un programme (en langage symbolique) dont la première ligne sera à l'adresse 10h de la mémoire, permettant d'ajouter deux nombres stockés aux adresses 01h et 02h et rangeant le résultat à l'adresse 03h.
3. Ecrire l'algorithme, puis le programme (commençant en 10h réalisant la multiplication (par additions successives) de deux nombres strictement positifs (N1 et N2) stockés en 01h et 02h et rangeant le résultat en 03h. (on supposera que les constantes 0 et 1 sont rangées respectivement en 05h et 06h). Représenter sous forme de tableau, l'évolution des registres ACCU, FLAGS et IP au cours de l'exécution du programme (en prenant N1=2 et N2=3).

Exercice 2

Comparez le style de la programmation sur des ordinateurs ayant respectivement des instructions à zéro(machine à pile), un, deux ou trois opérandes en écrivant quatre programmes qui calculent l'expression:

X = (A + B * C) / (D - E * F) .

Utilisez des instructions semblables à celles utilisées pendant le cours.

La comparaison portera sur la taille du programme et le trafic mémoire (recherche des opérandes)

Exercice 3

En utilisant la figure ci-après, donnez le contenu du registre r3 après l'exécution de chacune des instructions ou séquences d'instructions suivantes, prises individuellement. Le format générique de l'instruction ADD sera:

add src1,scr2,dst ! dst = scr1 + scr2

Le mode d'adressage s'applique au second opérande.

- add r1,r2,r3 ! mode registre
- add r1,r2(r1),r3 ! mode indexé
- add r1,@#adr,r3 ! mode direct avec adr = 1000_{16}
- add r1,#10,r3 ! mode immédiat
- add r1,@adr,r3 ! mode indirect avec adr = 1014_{16}

Les valeurs ci-après sont exprimées en hexadécimal.

r0	1014
r1	4
r2	1008
r3	C
r4	

Registres

1000	100C
1004	60
1008	3F3
100C	80
1010	20
1014	1004
1018	1018

Mémoire

Exercice 4 :

L'ordinateur dont il est question ici est une architecture machine à pile. Les mnémoniques à considérer, pour les instructions arithmétiques, sont :
ADD - SUB – MUL et DIV.
Donner le code nécessaire pour évaluer l'expression : F=(A+B)/((C-D)*E) .

Exercice 5 :

L'ordinateur dont il est ici question possède une architecture dont les instructions machines possèdent un seul opérande. Les mnémoniques à considérer, pour les instructions arithmétiques, sont ADD - SUB -MUL et DIV, et pour les instructions de manipulation de données LD et ST. Donner alors le code nécessaire pour évaluer l'expression : Z = (A-B/C)*(D+E) en utilisant le moins d'instructions possible.

Exercice 6 :

Voici la liste des instructions d'un ordinateur fictif :

4 instructions UAL :

- ADD Rd,Rs
- SUB Rd,Rs
- MULT Rd, Rs
- DIV Rd,Rs

2 instructions de chargement :

- LOAD Rd, [Rs]
- LOAD Rd, Immédiat

1 instruction de rangement : STORE [Rd], Rs

1 instruction de branchement : BEQZ Rs, Immédiat

L'ordinateur possède 8 registres 32 bits et les valeurs immédiates sont encodées sur 32 bits.

- Dans un premier temps, on utilise un seul format d'instruction de longueur fixe et à champ fixe. Énumérez les différents champs d'une instruction ainsi que le nombre de bits accordés à chaque champ. Quelle est la longueur totale d'une instruction ?
- Afin de réduire la longueur des programmes, on utilise deux formats d'instructions de longueur fixe à champ fixe. Décrivez ces deux formats.

Exercice 7 :

Un jeu d'instructions pour une architecture chargement-rangement utilise les deux formats d'instructions suivants :

Type d'instruction A (Rangement, chargement (fetch), branchements (branches) et sauts (jumps)) :

6 bits	4 bits	16 bits
Code-op	Rs/Rd	Immédiat

Type d'instruction B (Opérations UAL (ALU)) :

6 bits	4 bits	4 bits
Code-op	Rs	Rd

a) Déterminez le nombre de registres de cette architecture.
b) Déterminez le nombre de combinaisons Instructions/Modes d'adressages de cette architecture.
c) Une seconde alternative consiste à utiliser un seul format d'instruction de longueur fixe.Quelle sera la longueur (en bits) de ce format?

Exercice 8 : (contrôle n°2 2003)

On considère les 2 architectures suivantes : M1 une architecture à accumulateur et M2 une architecture à chargement/rangement (registre/registre) qui dispose de 16 registres . M1 et M2 disposent des instructions assembleur suivantes : LOAD , STORE et ADD
a) Détailler LOAD,STORE et ADD(càd spécifier le format: codop opérandes) pour M1 et M2
b) Ecrire une séquence de code assembleur pour M1 et M2 correspondant au programme de haut niveau suivant :

```
a := b + c ;
b := a + c ;
d := a + b ;
```

LE µPROCESSEUR 8086
UTILITAIRES DE BASE
REPERTOIRE D'INSTRUCTIONS

Développement d'un programme assembleur :

Le développement d'un programme depuis l'analyse du problème jusqu'à sa mise au point nécessite de nombreux outils logiciels qui constituent un environnement de programmation.
(l'environnement qui sera utilisé pendant ce cours est un environnement intégré appelé tasmedit).

Editeur de texte :

Un éditeur de texte est un logiciel interactif qui permet de saisir du texte à partir d'un clavier et de le stocker dans un fichier. Les informations contenues dans le fichier sont du type texte en code ASCII. Les principales fonctions d'un éditeur sont : la visualisation d'une partie du texte sur l'écran , le déplacement le positionnement du curseur , la modification du texte par insertion , suppression ou remplacement et la recherche de chaînes de caractères particulières.
Le programme source en assembleur est saisie à l'aide d'un éditeur de texte et aura pour extension asm

L'assembleur :

L'assembleur est un programme traducteur qui traduit le programme source assembleur en langage machine et le programme objet ainsi obtenu est rangé dans un fichier dont l'extension est obj (fichier objet). L'opération d'assemblage traduit chaque instruction du programme source en une instruction machine (code binaire).

Editeur de liens :

Le fichier .obj contient donc le produit binaire de l'assemblage mais le fichier est inutilisable tel quel , le système ne peut le charger et encore moins l'exécuter. En effet , il arrive fréquemment que l'on construit un programme exécutable à partir de plusieurs fichiers sources et donc il faut relier les fichiers objets à l'aide d'un programme nommé éditeur de liens (même si l'on en a qu'un seul).
L'éditeur de lien fabrique un fichier exécutable avec l'extension exe.

Chargeur :

On ne peut exécuter un programme que s'il se trouve en mémoire centrale , celui qui s'occupe de cette tache est appelé chargeur. Un utilitaire spécial du système d'exploitation (ici DOS) est responsable de la lecture du fichier exécutable , de son implantation en mémoire principale puis du lancement du programme.

Débogueur :

Le débogueur est un logiciel qui facilite la mise au point de programmes , il permet d'examiner le contenu des registres ainsi que le dumping de la mémoire. Ainsi on peut suivre l'exécution d'un programme pas à pas càd instruction après instruction et ceci permet de comprendre ce qui se passe lors de l'exécution.

La mise en œuvre de ces utilitaires se fera pendant les séances de travaux pratiques en utilisant TASMEDIT qui contient un éditeur de texte et qui encapsule l'appel à tasm et tlink , td est utilisé à part.

Présentation du 8086 :

C'est une machine à accumulateur multiple mais pas vraiment une architecture à registres généraux càd des registres équivalents car chacun des registres est généralement utilisé dans un but précis.
Le 8086 possède un bus d'adresses de 20 bits et un bus de données de 16 bits.

Les registres du 8086 :

Les registres se classent en 4 catégories :

- les registres généraux (16 bits)
- les registres de segment (16 bits)
- les registres d'offset (16 bits)
- le registre d'état (16 bits)

les registres généraux de données :

ils ne sont pas réservés à un usage très précis , ce sont en quelque sorte des registres à tout faire mais ils ont tous une fonction principale qui les caractérise .
AX (AH et AL) : accumulateur indispensable aux opérations de transferts et de calcul
BX (BH et BL) : base utilisé dans les calcul d'adresses
CX (CH et CL) : compteur il sert de compteur pour les boucles et les décalages
DX (DH et DL) : données à usage général

Les registres de segments :

Les registres de segment sont des registres à usages spéciaux , le 8086 utilise un ou plusieurs pour accéder à la mémoire et ces registres permettent au programmeur d'accéder simultanément à 4 zones d'adresses différentes.
CS : pointe sur la zone mémoire contenant les instructions du programme exécutable
DS : pointe sur la zone mémoire des données
ES : pointe sur une zone mémoire de données supplémentaires (utilise pour transfert bloc)
SS : ointe sur la zone mémoire pile qui est utile pour appel de procédures et passage de paramètres .

Les registres d'offset :

Ce sont des registres d'adressage à l'intérieur d'un segment c'est pour cela qu'on les appelle aussi registre décalage et ils ont pour fonction de marquer en permanence une position déterminée de la mémoire.
IP : pointeur d'instruction (compteur ordinal) il désigne le décalage de la prochaine instruction à exécuter par rapport au segment adresse par CS et il forme la paire de registre CS : IP
SP : pointeur de pile et désigne constamment la position du sommet de la pile SS : SP
BP : pointeur de base de la pile et il est utilisé pour accéder aux données de la pile lors d'appel de sous programme (transmission de paramètres par pile) SS : BP
SI : index source utilisé pour l'accès aux données DS : SI
DI : index destination utilisé pour l'accès aux données ES : DI

Le registre d'état :

Ce registre est un ensemble de 16 bits dont chacun représente une signification particulière et on les appelle des indicateurs (flags). D'une manière générale les flags fournissent des informations sur les résultats des opérations précédentes.
Bit 0 : CF (retenue)
Bit 2 : PF (parité)
Bit 4 : A (retenue auxiliaire)
Bit 6 : Z (zéro)
Bit 7 : S (signe)
.....