Institut
Mines-Télécom

Langage C et Systèmes d'exploitation

Introduction

Responsable : Etienne Borde (C218-3)

Auteur : Thomas Robert

Informations pratiques

- **Présence à chaque TH de cours, de TP, et de TD obligatoire**
  - Absence non-justifiée à 1 TH est tolérée
  - -1 point (sur la note finale) par TH avec absence non-justifiée

- **Evaluation de l'UE:**
  - 1 partiel sur le C et la chaine de production (40% de la note)
  - 1 CC sur le système en fin d'UE (60% de la note finale)

Rappel : un algorithme

- **Définition : description d'un ensemble d'opérations à réaliser sur des données**
- **Problème : les opérations utilisées dans la description de l'algorithme peuvent rester relativement abstraites**
- **=> Comment passe-t-on de cette description à sa réalisation ?**
- **Langage de programmation :**
  - une manière de référencer les données manipulées
  - leur représentation et structuration (e.g. tableaux, file …)
  - les opérations à appliquer aux données et leur impact

Un exemple d'algorithme avant de continuer…

- **Algorithme : calcul d'intersection de deux ensembles**
- **Types (mathématique) : ensemble (ENS)**
- **Inter(X,Y:ENS)**

{

$Z \leftarrow \emptyset$

pour chaque $x \in X$

{

$x \in Y \Rightarrow Z \leftarrow Z \cup \{x\}$

}

}

Affectation

Expression (logique)

Structure de contrôle d'exécution

? Et ça ?
Une sorte de
if then else ?

Savez vous répondre aux questions :

- Comment un ordinateur stocke les données ?
- Quelles opérations ***élémentaires*** sait faire un ordinateur ?
- Pourquoi passe-t-on par un langage de programmation ?
  *(pourquoi ne décrit on pas un programme en opérations **élémentaires**)*
- Quelles sont les étapes menant de la définition d'un programme à son exécution ?
- A quoi sert un système d'exploitation

Objectif : connaître les réponses et savoir les expliquer
(principe – limites/contraintes d'usage)

Quel est l'intérêt de connaître les réponses

- **Savoir ce qui faisable :** comprendre comment l'ordinateur fonctionne et va « exécuter vos programmes » permet de réaliser des applications plus complexes (en sachant ce qui est facile ou dur à réaliser)

- **Efficacité des traitements :** comprendre votre usage des ressources de calcul et stockage permet de faire des programmes plus efficaces

- **Se préparer à d'autres langages que C et JAVA** : ces concepts sont la base de l'informatique pratique, les connaître permet de se former plus facilement à de nouveaux langages

Suite de la séance

- **Introduction au concept de plateforme d'exécution**
  - Fonctionnalités et composants clés
  - Calculateur : principes de fonctionnement et exemple

- **De l'algorithme à son exécution**
  - Rappel des caractéristiques d'un langage
  - Stratégies d'exécution :
    compilation en instruction machine ou interprétation
  - Problème du chargement et concepts associés
  - Compilation

Support d'exécution une introduction

Fonctionnalités et exemples

- **Fonctions :**
  - Fournir les moyens de stocker de l'information
  - Fournir les moyens d'exécuter le traitement correspondant à un programme
  - Fournir des moyens d'interagir avec d'autres supports d'exécution ou l'environnement physique.

- **Exemples de plateformes différentes :**

Portables

Ordinateur de bord

Poste de travail

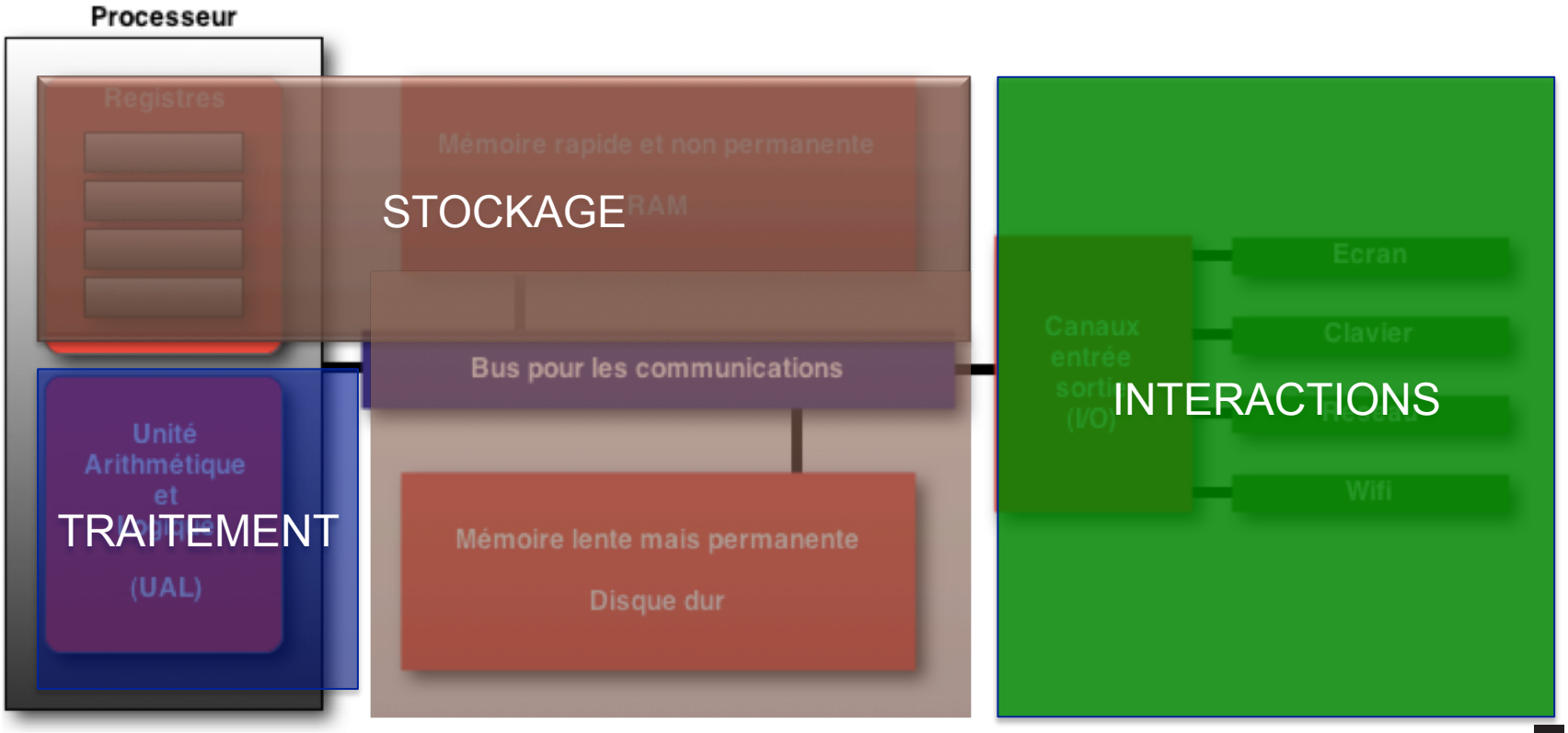
Un support d'exécution : les éléments clés

- **Un calculateur = regroupement de composants électroniques fournissant des capacités/ressources**
  - de stockage des données
  - de capture de stimuli extérieurs (physiques)
  - de communication de données avec d'autres calculateurs
  - de calcul et manipulation de données, contrôle d'exécution

- **Un système d'exploitation = logiciel pour organiser, simplifier et optimiser l'accès aux ressources**
  - Mise en place du partage et protection des ressources *(traitement / stockage)*
  - Simplification de l'usage et de l'accès aux ressources

Vision architecturale du calculateur

- **Intérêt : présenter une cartographie des ressources + de leurs interactions**



Stockage d'information

- **Un bit : unité de stockage d'information**
  peut prendre 2 valeurs : 0 et 1 (2 états possibles)
- **Les données manipulées par un processeur == séquences finies de bits**
  - 4 bits : 0110; 8 bits : 11010011…
- **Une séquence de N bits peut représenter n'importe quel élément d'un ensemble de taille inférieure à 2^N**
- **Ces séquences de bits peuvent être interprétées comme:**
  - Des nombres : 1001 = représentation base 2 de 9
  - Des mois : 000010000000 pour Mai

Une séquence de bits =
CE QUE VOUS DECIDEZ D'EN FAIRE (INTERPRETATION)

Un petit mot sur les systèmes numériques et le principe de représentation….

- **Séquence N bits interprétable comme un nombre en base 2**
- **ATTENTION : cela ne veut pas dire que l'on stocke toujours un nombre en mémoire**

  000010000000 voulait dire « Mai » et pas 2^7

⇒ **Indiquer à l'ordinateur comment interpréter les bits**

- **Représentations classiques :**
  - Système binaire : $b_7b_6….b_0$ ➔ $\sum_{i=0}^{7} b_i \cdot 2^i$
  - Complément à 2 : $b_7b_6….b_0$ ➔ $-(b_7 \cdot 2^7) + \left(\sum_{i=0}^{6} b_i \cdot 2^i \right)$
  - Symbole ASCII : des symboles « % » « A » « 7 » http://en.wikipedia.org/wiki/ASCII

Organisation du stockage

- **Organisation des bits en octet = 8 bits (granularité usuelle), plus pratique à écrire en base 16 (symboles: 0123456789ABCDEF)**

- **Problème du volume :**
  - Mozilla : 178 Mo de données utilisées dans certains cas
  - Base de données client : ~quelques Téra octets

Comment retrouver les données intéressantes dans un tel volume ?

- **Index et référence :**
  - Index : une fonction associant un identifiant à une donnée (mieux quand il est unique)
  - Référence : une valeur d'identifiant pouvant être utilisé pour récupérer la donnée qui lui est associée via l'index.
  - résoudre la référence = Index(Référence) →obtenir la valeur

Exemple Index – Référence

- **Identification des individus**
  - Index : Relation Identité – Numéro de sécurité sociale
  - Référence à une personne : 19303301213 77 (n° sécu)
- **Identification des données sur le « web » :**
  - Index : Relation URL – Donnée sur un serveur sur le web
  - Référence à une image : URL http://www.ici.fr/img.jpg
- **Identification d'un lieu (habitation)**
  - Index : Adresse Postale – Position géographique (lat,long)
  - Référence : 1 rue du lac, 75013 Paris

En pratique (1) mémoire

- **Mémoire (RAM) : composant permettant de stocker un ensemble d'octets de grande taille multiple d'une puissance de 2^K**

- **Caractéristiques :**
  - Index **:** 1 relation associant 1 numéro à chaque emplacement pouvant mémoriser 1 octet
    (le numéro est appelé **adresse**)
  - Granularité d'accès : groupe de 1 à 8 octets d'index consécutifs
    (ce groupe est appelé **mot mémoire)**
  - Référence à **un octet : une adresse**
  - Référence à **plusieurs octets contigus :**
    1 adresse @ + nombre d'octets à lire à partir de @

En pratique (2) Les autres stockages

- **Justification technologique de leur intérêt (et/ou):**

  Plus rapide, moins couteux utilisable directement par le processeur …
  beaucoup de raisons très variées

- **Registres : stockage mémorisant quelques octets**
  - taille = mot mémoire (8 octets sur une machine 64 bits)
  - Directement nommés => pas d'index à base de numéros
  - là où le processeur réalise ses calculs
- **Stockage de masse (disque dur) :**
  - Accès = bloc pour transferts vers ou depuis la mémoire (ensemble d'octet ~ 1000 octets)
  - Index structuré : (cas disque dur non SSD)
    - Plateau, cylindre … ~ adresse postale: rue, ville, pays

Bilan sur l'organisation du stockage

- **3 types de stockages essentiellement :**
  Registres, Mémoire, Stockage de masse (disque dur)

- **Notion d'index et de référence**
  - Index : le système d'association (id, valeur), l'id peut être défini en plusieurs morceaux
  - Référence : un id (permettant de retrouver une valeur si il est complet et à jour…)

- **Références aux données en mémoire par le processeur**
  - Index: adresse → 1 octet
  - Référence: dépend de ce qu'on veut récupérer… mais repose a priori sur le concept d'adresse

Capacités de traitement (1)

- **Composant clé : le processeur**
- **Traitements = instructions (opérations *élémentaires*)**
- **Jeu d'instructions = {des opérations élémentaires supportées par le processeur}**
  - Calculs numériques
  - Transferts de données entre supports de stockage
  - Evaluations de conditions booléennes

    ….

- **Problème: Comment indiquer au processeur les instructions à exécuter ?**

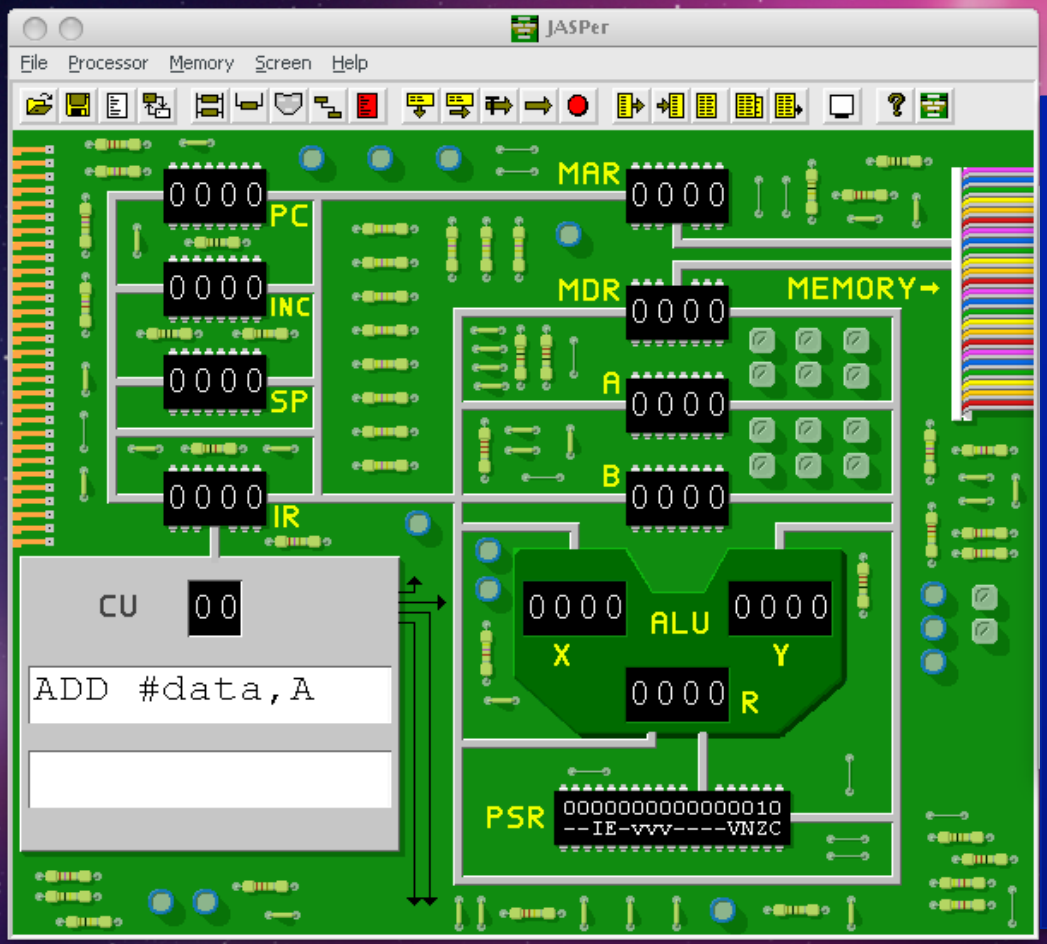
Capacités de traitement (2) : Un peu d'histoire… ça commence en 1801

- **Ancêtre de l'ordinateur : Métier à tisser « Jacquart »**
- **Objectif : tisser un motif à partir de fil de couleurs différentes**
- **Principe : automatisation des mouvements de crochets/navette**
- **Mise en œuvre :**
  - Carton perforé = représentation de la description des mouvements
  - Répétition du cycle :
    - Décoder le mouvement suivant (navette et crochets) 1 pour chaque
    - Opérer les mouvements décodés

Capacités de traitement (3)
Rappel Architecture Von Neumann

- **Caractéristiques principales**
  - Représentation binaire des instructions en mémoire
  - Stockage sur les mêmes supports des données ET de la **représentation** des instructions
- **Automatisation de l'exécution**
  - Compteur ordinal (CO): registre contenant l'adresse de la prochaine instruction
  - Répétition systématique de la logique d'exécution
    - Récupérer l'instruction associée à la référence contenue dans CO
    - Déterminer la nature de l'opération décrite par l'instruction
    - Récupérer les données d'entrée de l'opération

Représentation visuelle du fonctionnement d'un calculateur (processeur + mémoire) [1]



A trier

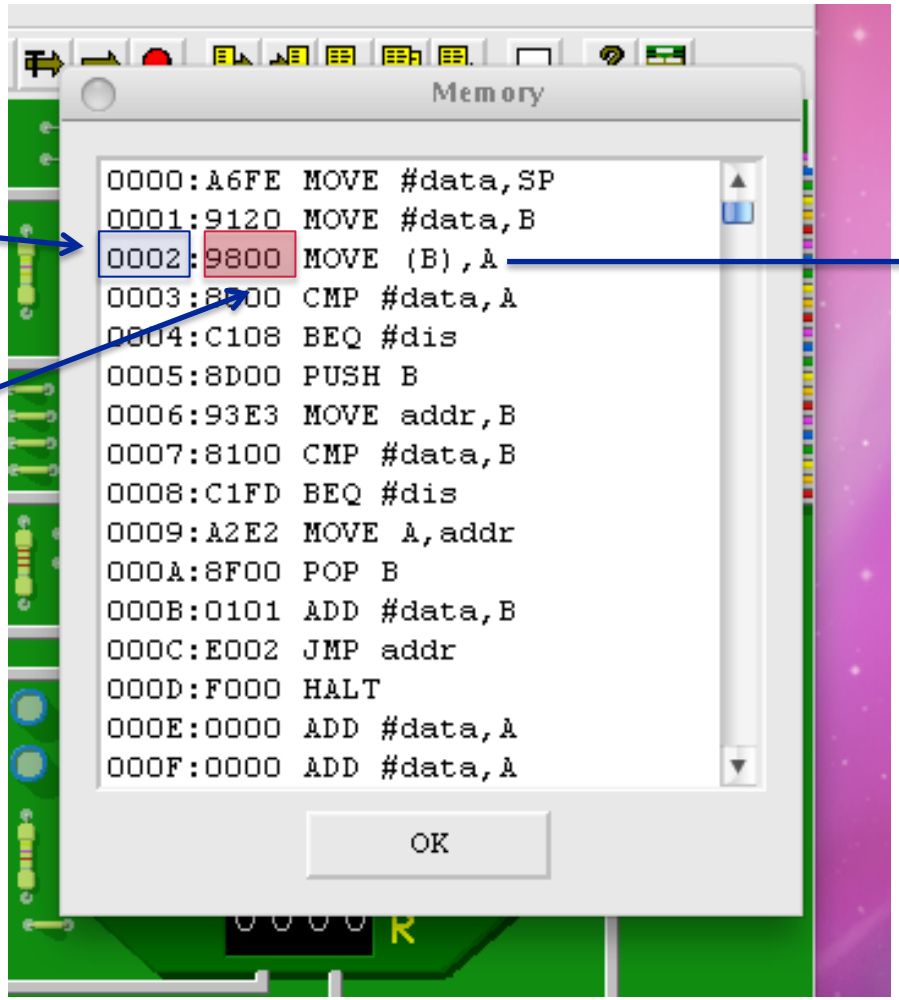
.DS_Store

Structure du processeur

- **Registres : PC,SP,IR,A,B**
- **Unité de contrôle (CU)**
  - Interprète IR
  - Orchestre les transferts inter registres
- **Mémoire**
  - Mots de 2 octets
  - Référence dans MAR
  - Valeur dans MDR
- **Codage instructions**:
  IR = 00 00 => A reçoit A +0
  - 2 premiers = opération
  - 2 derniers = données

Représentation instructions en mémoire

Référence = adresse



Interprétation

Valeur = Code instruction

ATTENTION
notation base 16
0123456789ABDEF

Instruction et comportements modifiant CO

- **Instruction de contrôle : instruction ayant un impact sur la valeur de CO**
  - Branchement de déroutement et restauration (paire d'instructions call-ret, int-iret…)
  - Branchement conditionnel (selon une condition)
  - Branchement inconditionnel (systématique)

- **Observation : possibilité de concevoir une séquence d'instructions**
  - Transférant des données entre registre et mémoire
  - Réalisant des calculs / comparaison
  - Déterminant l'adresse de la prochaine instruction à exécuter

Le cas particulier du Déroutement avec restauration

On veut calculer

$$\sum_{j=121}^{242} j$$

Rappel Somme 1…N = N*(N-1)/2
=> calcul possible en O(1)

idée N dans Ra et résultat dans Rb
Rb doit contenir ((Ra-1) * Ra)/2

Pb pas calculable en 1 coup

@	Instruction
911	Rb ← Ra-1
912	Rb ← Ra*Rb
913	Rb ← Rb/2
914	….
915	….

Le cas particulier du Déroutement avec restauration

On veut calculer

$$\sum_{j=121}^{242} j$$

Rappel Somme 1…N = N*(N-1)/2
=> calcul possible en O(1)

idée N dans Ra et résultat dans Rb
Rb doit contenir ((Ra-1) * Ra)/2

Pb pas calculable en 1 coup

@	Instruction
1	Ra ← 120
2	Call 911
3	Rc ← Rb
4	Ra← 242
5	Call 911
6	Rb ← Rb -Rc

@	Instruction
911	Rb ← Ra-1
912	Rb ← Ra*Rb
913	Rb ← Rb/2
914	Ret
915	….

Bilan sur les capacités d'exécution

- **Représentation en mémoire de séquence d'opérations dont l'exécution est automatisée**
- **Capacité à définir la prochaine instruction à exécuter**
- **Capacité à réutiliser une même séquence d'instructions dans différents « contextes »**
- **Capacité à pouvoir arrêter une séquence d'exécution pour faire autre chose (expliqué en cours de système):**
  - Soit parce que le processeur ne sait pas continuer
  - Soit parce qu'il faut exécuter un autre traitement (qui peut n'avoir aucun rapport avec celui en cours).

Rappel :
Un support d'exécution : les éléments clés

- **Un calculateur = regroupement de composants électroniques fournissant des capacités/ressources**
  - de stockage des données
  - de capture de stimuli extérieurs (physiques)
  - de communication de données avec d'autres calculateurs
  - de calcul et manipulation de données, contrôle d'exécution

Problème : coder directement en langage machine
== construire un château de sable grain par grain…

faisable mais très fastidieux et technique

Voyons l'intérêt des langages de programmation

De l’algorithme à son exécution

Langage de programmation
une convention d'écriture sur …

- **la définition des données (types, identifiants, visibilité, valeur..) et la manière d'y faire référence**

  int i=0; ou Eleve E;

- **les opérations que l'on s'autorise à utiliser sans avoir à en redéfinir systématiquement le sens**

  w← 3; (par exemple); ou « if (x<2) {x++;} »

- **la manière dont on définit des séquences d'instructions réutilisables (fonctions, méthodes) et leurs modes d'accès**

  this.set_x(4); vs set(x,4);

- **La structuration du programme**

  Classe / package / fichiers / modules et espaces de noms…

Pb de l'exécution d'un programme

- **Les éléments clés qui ont été abstraits :**
  - La nature des traitements et **leurs références**
  - La représentation des données et **leurs références**

- **Hypothèse : programme accessible en mémoire**

PB n°1 : la nature du traitement est décrit en texte et pas en instructions machine

PB n°2 : la représentation des données et de leurs références ne correspondent pas directement à des adresses

Stratégie de mise en œuvre

- **Interprétation (1 étape):**
  1. un ensemble d'instructions déjà en mémoire (le shell)
  - lit le programme (texte sous forme de séquence de commandes),
  - analyse les références (chemin vers un fichier),
  - réalise les traitements pas à pas

  Conséquence: analyse à refaire à chaque nouvelle exécution

- **Compilation et chargement pour exécution (2 étapes):**
  1. Compilation: traduit le programme (texte en C) en éléments équivalents du langage machine (adresses, instructions binaires)
  2. lancement de l'exécutable

  Conséquence: compilation à refaire à chaque modification du programme

  Un ensemble d'instructions (système d'exploitation) place la représentation binaire en mémoire pour en exécuter les instructions **(chargement + exécution du point d'entrée)**

Des exemples de langages et d'usages associés !!!

- **Programme à interpréter = script (ref. cinéma)**
  - Calculs et manipulation de textes : Lisp, perl
  - Langages web : PHP, scripts html
- **Langages compilés :**
  - C / C++ : programmation système / interface / robotique (beaucoup de programme en robotique en C++)
  - Ada : programmation systèmes critique (trains, avions, satellites…)
- **Ceux qui sont au milieu…**
  - Java : compilation des références et opérations complexes puis interprétation d'instruction génériques

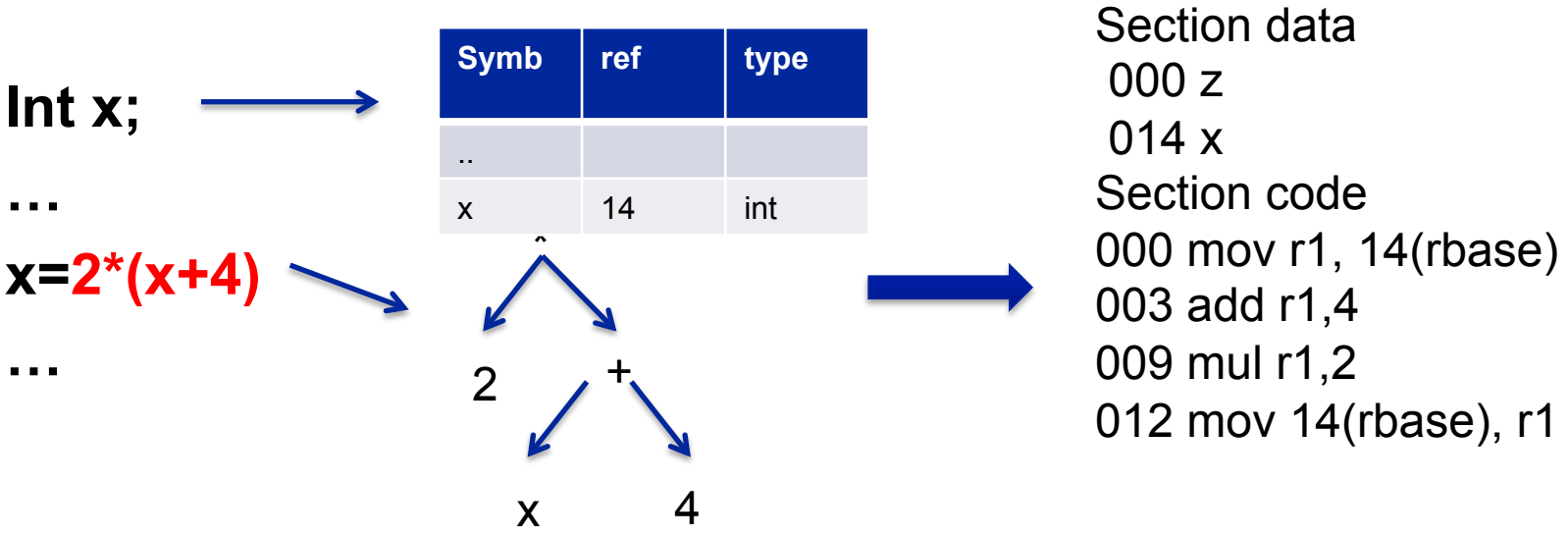
Principe de fonctionnement d'un compilateur (vous en verrez plus plus tard…)

- **Hypothèse : texte = succession de symboles**
- **Analyse lexicale: analyse des symboles pour identifier des mots clé, la ponctuation…**
- **Analyse syntaxique: identification de la structure des « phrases »**
  - sujet / verbe / compléments
  - <identifiant> <affectation> <expression>
- **Analyse sémantique: Interprétation du sens de la « phrase » par rapport à sa structure**
  - Apprend! Apprenez! Apprendriez vous?
  - Déclaration de variables, d'une opération, exécution d'un calcul…
- **Génération de code: synthèse du code machine équivalent / traduction des références….**

Compilation: traduction de références abstraites en références concrètes (adresses)

- **Table des symboles et adresses relatives**
  - Données et instructions organisées en sections
  - Rangées séquentiellement, avec un index relatif à la section
  - Référence concrète=adresse début de section+adresse relative

  Si un nom est utilisé mais pas défini => erreur de compilation
- **Compilation des expressions,**



Comment exécuter les programmes compilés
Problème du chargement

- **Compilation : traduction des concepts du programmes (donnée et références) en langage machine**
- **Comment l'exécuter?**
- **Idée : Si on a la représentation en langage machine il suffit :**
  - De la charger à une adresse X (copier le binaire en mémoire)
  - Calculer l'adresse du point d'entrée (adresse de la fonction main)
  - Modifier CO à cette valeur … c'est parti

Bilan langages de programmation : Du programme jusqu'à l'exécution

- **Le rôle de la compilation**
  - Traduction des opérations en séquences plus élémentaires
  - Traduction des références aux données et traitements
- **Le problème de la traduction des références**
  - Toutes les références n'ont pas une adresse fixe connue à l'avance => Que faire ?
  - Que faire si l'on ne souhaite pas placer les données toujours au même endroit ?

PB n° 1 : adresses absolues vs relatives

- **Programme :**

```
sommeKaN(K,N) {
  return (somme1aN(N)-somme1aN(K-1));
}

Somme1aN(N) {
  return (N*(N-1))/2;
}
```

@	Instruction
1	Ra ← 121
2	Call 911
3	Rc ← Rb
4	Ra← 242
5	Call 911
6	Rb ← Rb -Rc

Question : Que fait on si 911-915 est déjà occupé par d'autres instructions ?

Service Système n°1 : créer l'illusion que chaque application a son propre espace d'adressage.

PB n° 2 : Rappel convention pour fournir les données d'entrées : point d'entrée et paramètres

- **Ce que vous devez savoir :**
  - point d'entrée == **convention du langage de programmation** pour dire où l'exécution du programme doit débuter (*main (…) en java)*
  - Noms de variables prédéfinies contenant des données définies au moment du démarrage de l'exécution du programme (*args en Java*)

Question : qui configure le calculateur pour exécuter le programme (définition de CO == adresse de point d'entrée) ? qui récupère les valeurs des paramètres ?

Service Système n°2 : lancement du programme et résolution des références aux paramètres par un code tiers (crt0.o)

Vue simplifiée du chargement / exécution

- **Charger chaque « section en mémoire »**
- **Définir les registres d'adresses de début de section**
- **Calculer l'adresse du point d'entrée**
- **Placer les valeurs des paramètres à leur place attendue en mémoire**
- **Modifier CO pour exécuter le point d'entrée**

…. Exécution du programme ….

Et avec tout cela on fait quoi

- **Presque tout ce que vous utilisez tous les jours**
  - Systèmes bureautiques
  - Tableau de contrôle d'un véhicule
  - Tableau de commande d'un robot outil
  - Serveur de transactions bancaires
- **Les concepts qu'on abordera en détail dans l'UE:**
  - Les constructions du langages C : principes et usages
  - La pratique de la compilation : le C en exemple
  - Le fonctionnement des trois service clés du SE : ordonnancement, gestion mémoire, et synchronisation
  - Et un peu de système de fichier …