Introduction

IFT209 – Programmation système (gr. 01)

Hiver 2024

Aujourd'hui

- Plan de cours
- Introduction

Plan de cours

Enseignant

Michael Blondin

@ michael.blondin@usherbrooke.ca

info.usherbrooke.ca/mblondin

bureau D4-1024-1 au 1er étage

Horaire

Cours magistraux	Laboratoires
Mercredi	Jeudi
13h30 – 15h20	08h30 – 10h20
D3-2032	Laboratoire

Horaire

Cours magistraux	Laboratoires
Mercredi	Jeudi
13h30 – 15h20	08h30 – 10h20
D3-2032	Laboratoire *

* Cours: **11 jan.**, 25 jan., 22 fév., 11 avr. au D4-2025

But du cours

- Comprendre l'architecture d'un ordinateur
- Connaître les types élémentaires de données et leur représentation
- Savoir manipuler les données en mémoire
- Se familiariser avec les langages de bas de niveau
- Améliorer l'aisance générale en programmation

Préalables

Préalable:

- IFT159 – Analyse et programmation

Préalable à:

- IFT320 – Systèmes d'exploitation
- IFT585 – Télématique

Utile pour:

- IFT580 – Compilation et interprétation des langages

Contenu

1. Systèmes de numération
2. Prog. en « assembleur »
3. Architecture des ordinateurs
4. Nombres entiers
5. Accès aux données
6. Tableaux
7. Circuits logiques
7. Prog. structurée
8. Chaînes de bits
9. Chaînes de caractères
10. Sous-prog. et mémoire
11. Nombres en virgule flottante
12. Entrées/sorties

Contenu

1. Systèmes de numération **2h**

- Écriture de nombres dans un système de numération
- Systèmes binaire, décimal et hexadécimal
- Conversion de nombres entre systèmes
- Arithmétique

Contenu

2. Programmation en langage d'assemblage **2h + 2h**

- Survol à partir de courts programmes
- Introduction à un jeu d'instructions
- Programmation de haut niveau des entrées/sorties

Contenu

3. Architecture des ordinateurs **2h**

- Architecture de von Neumann
- Mémoire principale, processeur et registres
- Jeux d'instructions
- Organisation d'un ordinateur

Contenu

4. Nombres entiers — 2h + 2h

- Représentation des entiers signés/non signés
- Report et débordement
- Instructions arithmétiques

Contenu

5. Accès aux données **1h**

- Données
- Adresses
- Modes d’adressage
- Étapes de la vie d’un programme

Contenu

6. Tableaux **1h + 2h**

- Tableaux à une, deux ou plusieurs dimensions
- Allocation et initialisation
- Parcours

Contenu

7. Programmation structurée **2h + 2h**

- Structures de contrôle
- Condition et branchement
- Appel et retour de sous-programmes

Contenu

8. Circuits logiques **2h**

- Arithmétique
- Décodage
- Mémoire

Contenu

9. Chaînes de bits — 2h + 2h

- Algèbre de Boole et valeurs booléennes
- Opérations logiques
- Décalages de bits
- Masquage

Contenu

10. Chaînes de caractères **1h**

- Représentation de caractères
- ASCII, ISO 8859-1, UTF-8/16/32
- Opérations sur les (chaînes de) caractères

Contenu

11. Sous-programmes et mémoire **1h**

- Disposition de la mémoire
- Appel et retour de sous-programmes
- Passage de paramètres
- Pile d'exécution: sauvegarde et récupération
- Récursivité

Contenu

12. Nombres en virgule flottante — 2h + 2h

- Représentations des nombres en virgule flottante
- Erreurs d'arrondi et de troncation
- Dépassement de capacité
- Norme IEEE 754
- Instructions arithmétiques

Contenu

13. Entrées/sorties **4h**

- Mécanismes de gestion des entrées/sorties
- Interruptions et leur traitement
- Illustration à l'aide de dispositifs simples

Matériel

- Aujourd'hui: premier et dernier cours avec diaporama
- **Notes électroniques** complètes déjà en ligne

Évaluation

Laboratoires	12% ($6 \times 2\%$)
Devoirs	28% ($2 \times 5\%, 3 \times 6\%$)
Examen périodique	25%
Examen final	35%

Devoirs

- 5 devoirs
- À faire en équipes de deux
- Premier: $\sim$2 sem. (manuscrit)
- Autres: 10 jours à 3 sem. (programmation)

Laboratoires

- 6 laboratoires
- À faire en équipes de deux
- Échéancier:

Affichés:	mercredi	à 13h30
En classe:	jeudi	à 08h30
À remettre:	dimanche	à 23h59

Rétroaction non officielle après l'examen périodique

Calendrier

Sujets	Laboratoire	Devoirs
1: Intro., sys. numération	Cours	Devoir 1 ∼13 jours
2: Prog. en lang. d'assemblage	Labo 1	Devoir 1 ∼14 jours
3: Architecture, nombre entiers	Cours	
4: Levée de cours	Labo 2	
5: Accès aux données et tableaux	Labo 3	Devoir 3 ∼14 jours
6: Prog. structurée	Labo 4	
7: Circuits logiques	Révision	—
8: Examen périodique	—	—
9: Relâche	—	—

Calendrier

Sujets	Laboratoire	Devoirs
10: Retour examem, manip. bits	Labo 5	Devoir 4 ∼21 jours
11: Chaînes de car., sous-prog.	Travail devoir 4	
12: Nombres en virg. flottantes	Labo 6	
13: Entrées/sorties	Travail devoir 5	Devoir 5 ∼10 jours
14: Entrées/sorties	Révision	
15: Examen final	—	—
16: Examen final	—	—

Disponibilités

Sur **rendez-vous** à mon bureau

et

1h / semaine (à choisir maintenant)

Page Web

- mblondin.espaceweb.usherbrooke.ca/ift209

Introduction

Programmes

```
int main()
{
}
```

```
main:
.LFB0:
    .cfi_startproc
    xorl   %eax, %eax
    ret
    .cfi_endproc
    ; ...
```

```
46 E3 0D
DF 62 2A
1A 3F B0
   ...
```

```
1
0
0
0
1
1
0
1
⋮
```

haut niveau → **bas niveau** → **code machine**

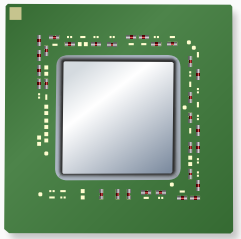
Programmes

code machine

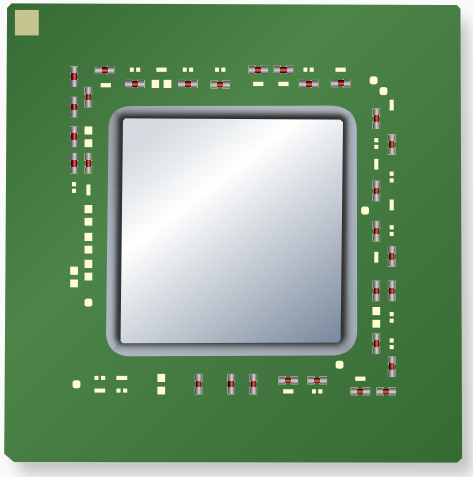
1
0
0
0
0
1
1
0
1
⋮

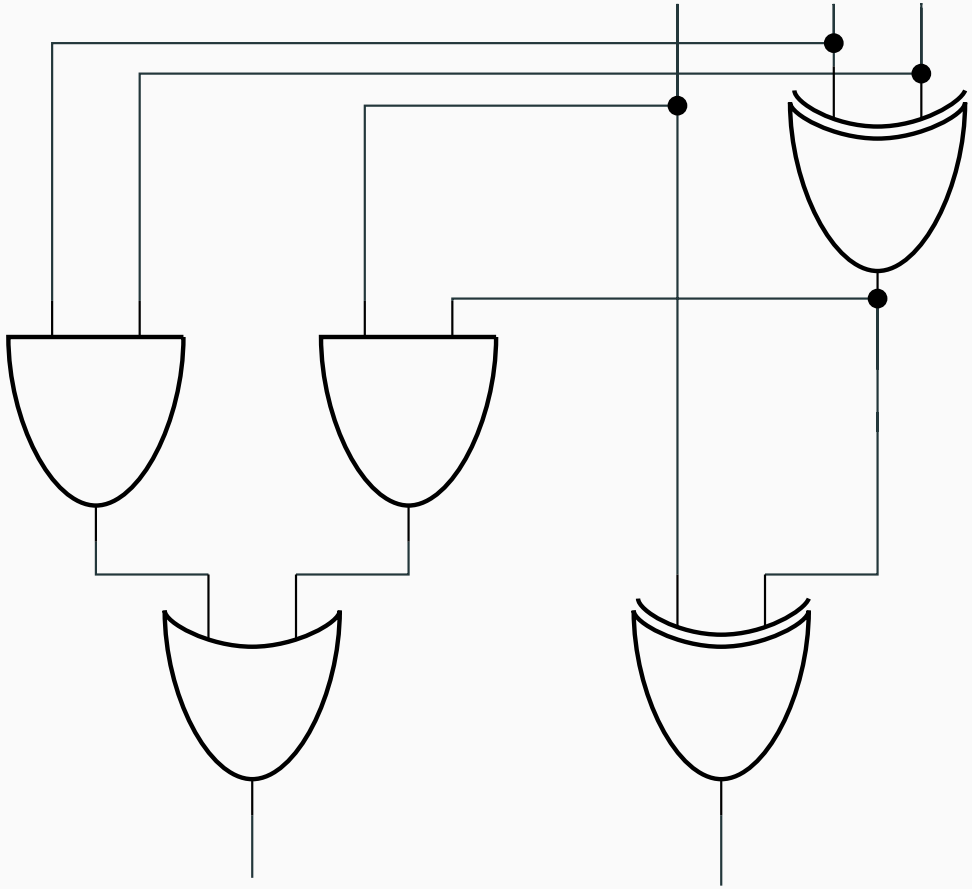
sys. d'exploitation

processeur



Programmes





processeur → **circuits**

Programmes

haut niveau (la plupart des cours)

➔ **bas niveau**

code machine

sys. d'exploitation (IFT320)

processeur et architecture

circuits

Programmation de bas niveau

- Bas niveau: *langages d'assemblage*
- Varie selon l'architecture matérielle
- Possible de coder un programme complet!

Programmation de bas niveau

- Bas niveau: *langages d'assemblage*
- Varie selon l'architecture matérielle
- Possible de coder un programme complet!

© *Spacewar!* (1962)

```
tno,  6,        law i 41
tvl,  7,        sar 4s
rlt, 10,        law i 20
tlf, 11,        law i 140
foo, 12,        -20000
maa, 13,        10
sac, 14,        sar 4s
str, 15,        1
me1, 16,        6000
...
```

Programmation de bas niveau

- Bas niveau: *langages d'assemblage*
- Varie selon l'architecture matérielle
- Possible de coder un programme complet!

La plupart des ordinateurs personnels:

x86-64




Majorité du cours:

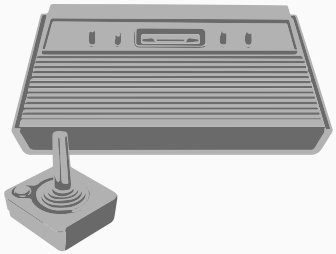
ARMv8 (AArch64/ARM64)

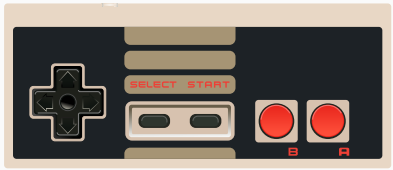
Majorité du cours:

ARMv8 (AArch64/ARM64)

Fin du cours (entrées/sorties):

NMOS 6502





Aujourd'hui seulement:

(architecture ouverte et libre)

Survol

Que représente cette séquence?

```
01000001011011000110110010010011
```

(a) un entier?

(b) un nombre en virgule flottante?

(c) une chaîne de caractères?

(d) un tableau?

(e) un programme?

Que représente cette séquence?

```
01000001011011000110110010010011
```

Entier	`1097624723`
Nombre en virg. flot.	`14.776507`
Chaîne	`"Allô"` ou *invalide*

Tableau	`[165, 108, 108, 147]`
	`[165, 108, 108, -109]`
	`[16748, 27795]`
	`[1097624723]` + leurs versions 2D

Programme (ARMv8)	`sbfiz x1, x2, 0x14, 0x1C`

Programme (x86-64)	`rex.B ins BYTE PTR es:[rdi],dx`
	`ins BYTE PTR es:[rdi],dx`
	`xchg ebx,eax`

Que représente cette séquence?

```
01000001011011000110110010010011
```

Tout est binaire,

les types sont un construit...

Que représente cette séquence?

```
01000001011011000110110010010011
```

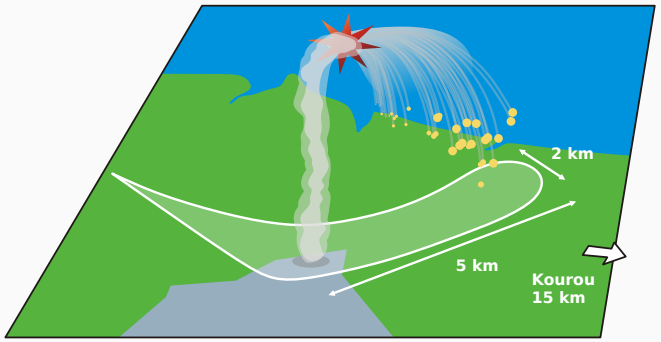
Tout est binaire,

les types sont un construit...

(En réalité: tout est continu...)

Survol

Il faut les manipuler avec prudence! (Ariane 5)



Tout est binaire,

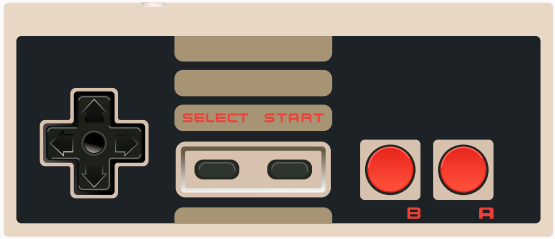
les types sont un construit...

Survol

Tout est binaire,

les types sont un construit...

Devoir 5



Tout est binaire,

les types sont un construit...

Survol

Que fait ce programme? Comment est-il compilé? Et exécuté?

```
unsigned long foo(unsigned long n)
{
  unsigned long t = 0;

  while (n > 0) {
    t += n;
    n -= 1;
  }

  return t;
}
```

Comment calculer la suite de Fibonacci?

n	`fib(`n`)`
0	0
1	1
2	1
3	2
4	3
5	5
6	8
⋮	⋮

Survol

Que se produit-t-il si on exécute ces programmes?

```
void foo()
{
  while (true) { }
}
```

```
void bar()
{
  bar();
}
```

Survol

Combien d'itérations effectuent ces programmes?

```
float y = 0.0;

while (y != 1.0) {
  y += 0.25;
}
```

```
float x = 0.0;

while (x != 1.0) {
  x += 0.1;
}
```

Survol

Qu'est-ce que ce programme affiche?

```c
char x[] = "foo";
char y[] = "bar";
char z[] = "bonjour";

for (size_t i = 0; i < sizeof(z); i++) {
  x[i] = z[i];
}

printf("%s\n", x);
printf("%s\n", y);
printf("%s\n", z);
```

Questions?

À demain!