

Introduction

IFT209 – Programmation système

Hiver 2022



- Plan de cours
- Introduction

Plan de cours

Michael Blondin

@ michael.blondin@usherbrooke.ca

 info.usherbrooke.ca/mbлондин

 bureau D4-1024-1 au 1^{er} étage

Cours magistraux	Laboratoires
Lundi	Jeudi
13h30 – 15h20	08h30 – 10h20
Teams (D7 -2023)	Laboratoire

Cours magistraux	Laboratoires
Lundi	Jeudi
13h30 – 15h20	08h30 – 10h20
Teams (D7-2023)	Laboratoire *

* Cours: **6 jan.**, 13 jan., 17 fév. et 14 avr. (~~D7-2023~~)

- 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

Préalable:

- IFT159 – Analyse et programmation

Préalable à:

- IFT320 – Systèmes d'exploitation
- IFT585 – Télématique

Utile pour:

- IFT580 – Compilation et interprétation des langages

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

1. Systèmes de numération

2h

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

2. Architecture des ordinateurs

2h

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

3. 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

4. Accès aux données

2h

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

5. Nombres entiers

2h + 2h

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

6. Tableaux

2h + 2h

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

7. Circuits logiques

2h

- Arithmétique
- Décodage
- Mémoire

8. Programmation structurée

2h + 2h

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

9. Chaînes de bits

2h + 2h

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

10. Chaînes de caractères

1h

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

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

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

13. Entrées/sorties

4h

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

- Aujourd'hui: premier et dernier cours avec diaporama
- **Notes électroniques** mises à jour chaque semaine
- Référence complémentaire:

RICHARD ST-DENIS: *L'architecture du processeur SPARC et sa programmation en langage d'assemblage*

Laboratoires	12% ($6 \times 2\%$)
Devoirs	30% ($5 \times 6\%$)
Examen périodique	24%
Examen final	34%

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

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

Affichés: mercredi à 08h30

En classe: jeudi à 08h30

À remettre: dimanche à 23h59

Rétroaction non officielle après
l'examen périodique

Sujets	Laboratoire	Devoirs
1: Introduction	Cours	
2: Sys. numération, architecture	Cours	Devoir 1 ~10 jours
3: Prog. en lang. d'assemblage	Labo 1	Devoir 2 ~14 jours
4: Accès aux données	Travail devoir 2	
5: Entiers	Labo 2	Devoir 3 ~14 jours
6: Tableaux	Labo 3	
7: Circuits logiques	Révision	—
8: Examen périodique	—	—
9: Relâche	—	—

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

Sur **rendez-vous** sur Teams (~~à mon bureau~~)

et

1h / semaine (à choisir maintenant)

 info.usherbrooke.ca/mblondin/cours/ift209_h22

Introduction

			1
	main:		0
	.LFB0:		0
int main()	.cfi_startproc	46 E3 0D	0
{	xorl %eax, %eax	DF 62 2A	1
}	ret	1A 3F B0	1
	.cfi_endproc	...	0
	; ...		1
			:

haut niveau



bas niveau



code machine

code machine

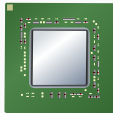
1
0
0
0
1
1
0
1
:
:
:

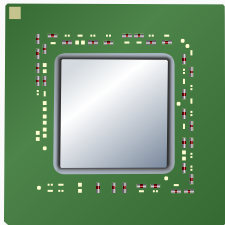


sys. d'exploitation

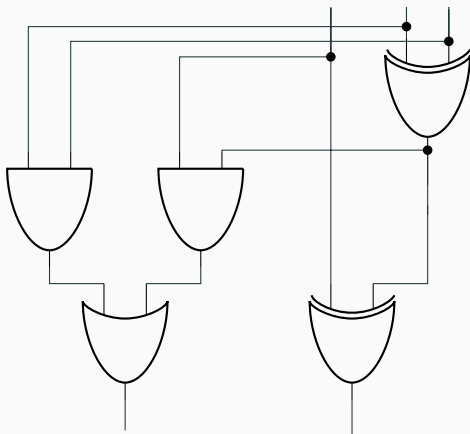


processeur





processeur



circuits

haut niveau (la plupart des cours)

➔ **bas niveau**

code machine

sys. d'exploitation (IFT320)

processeur et architecture

circuits

- 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!



Votre ordinateur:

x86-64



Majorité du cours:

ARMv8 (AArch64/ARM64)



Majorité du cours:

ARMv8 (AArch64/ARM64)

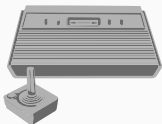


The screenshot shows the Droid Info application interface. At the top, there's a status bar with 'Koodo', signal strength, 55% battery, and 14:47. Below is a blue header with the Droid Info logo and title. A navigation bar contains three tabs: 'Dispositif', 'Système' (which is selected and highlighted in blue), and 'Mémoire'. The main content area is titled 'PROCESSEUR' in a dark bar. Below this, various system specifications are listed in a table-like format with labels on the left and values on the right, many of which are hyperlinks.

PROCESSEUR	
Architecture du CPU	ARMv8-A
Board	EML
Chipset	KIRIN970
Cœurs	8
Vitesse d'horloge	1690 MHz - 2362 MHz
Sets d'instructions	arm64-v8a
Caractéristiques du CPU	fp asimd evtstrm aes pmull sha1 sha2 crc32
Gouverneur du CPU	interactive
Version du noyau	4.4.103+
Architecture du noyau	aarch64

Fin du cours (entrées/sorties):

NMOS 6502



Aujourd'hui seulement:



(architecture ouverte et libre)

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 <i>invalid</i>
Tableau	[165, 108, 108, 147] [165, 108, 108, -109] [16748, 27795] [1097624723] + leurs versions 2D
Programme (ARMv8)	<code>sbfiz x1, x2, 0x14, 0x1C</code>
Programme (x86-64)	<code>rex.B ins BYTE PTR es:[rdi],dx</code> <code>ins BYTE PTR es:[rdi],dx</code> <code>xchg ebx,eax</code>

Que représente cette séquence?

01000001011011000110110010010011

*Tout est binaire,
les types sont un construit...*



Que représente cette séquence?

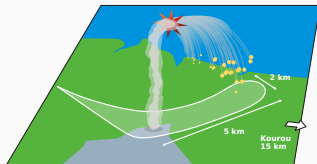
01000001011011000110110010010011

*Tout est binaire,
les types sont un construit...*

(En réalité: tout est continu...)



Il faut les manipuler avec prudence! (Ariane 5)



*Tout est binaire,
les types sont un construit...*





*Tout est binaire,
les types sont un construit...*



Devoir 5



*Tout est binaire,
les types sont un construit...*



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	$\text{fib}(n)$
0	0
1	1
2	1
3	2
4	3
5	5
6	8
$\vdots$	$\vdots$

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

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

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

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;  
}
```

Qu'est-ce que ce programme affiche?

```
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 lundi!