

Introduction

IFT209 – Programmation système

Hiver 2021



- Plan de cours
- Introduction

Plan de cours

Michael Blondin

@ michael.blondin@usherbrooke.ca

 info.usherbrooke.ca/mblondin

 bureau D4-1024-1 au 1^{er} étage

Cours magistraux	Laboratoires
Lundi	Jeudi
13h00 – 14h50	08h00 – 09h50
Teams (D2-1060)	Laboratoire

Cours magistraux	Laboratoires
Lundi	Jeudi
13h00 – 14h50	08h00 – 09h50
Teams (D2-1060)	Laboratoire *

* Cours: **14 jan.**, 28 jan., 18 fév. et 8 avr. (D2-1060)

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

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. 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

3. Architecture des ordinateurs

2h

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

4. Accès aux données + circuits

2h

- Données
- Adresses
- Modes d'adressage
- Étapes de la vie d'un programme
- Survol des circuits logiques

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. Programmation structurée

2h

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

8. Chaînes de bits

2h + 2h

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

9. Chaînes de caractères

1h + 1h

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

10. Sous-programmes et mémoire

1h + 1h

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

11. 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

12. Entrées/sorties

4h + 2h

- 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	10% (5 × 2%)
Devoirs	30% (5 × 6%)
Examen périodique	25%
Examen final	35%

- 5 devoirs
- À faire en équipes de deux
- Premier: 1 semaine (manuscrit)
- Autres: \sim 2 semaines (programmation)

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

Affichés: mercredi à 08h00

En classe: jeudi à 08h00

À remettre: dimanche à 23h59

Rétroaction non officielle après
l'examen périodique

Sujets	Laboratoire	Devoirs
1: Introduction, sys. numération	Cours	Devoir 1 ~7 jours
2: Prog. en « assembleur »	Labo 1	
3: Architecture, données, circuits	Cours	Devoir 2 ~14 jours
4: Entiers	Labo 2	
5: Tableaux	Labo 3	Devoir 3 ~14 jours
6: Programmation structurée	Révision	
7: Examen périodique	—	—
8: Relâche	—	—

Sujets	Laboratoire	Devoirs
9: Retour examen, bits	Labo 4	Devoir 4 ~16 jours
10: Caractères, sous-prog.	Travail devoir 4	
11: Nombre virg. flottante	Labo 5	
12: Entrées/sorties	Travail devoir 5	Devoir 5 ~11 jours
13: Congé	Cours	
14: Révision	—	—
15: Examen final	—	—
16: Examen final	—	—

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

et

1h / semaine (à choisir maintenant)

 info.usherbrooke.ca/mblondin/ift209

Introduction

Programmes

			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

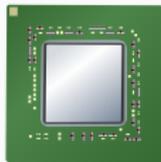
1
0
0
0
1
1
0
1
:
:

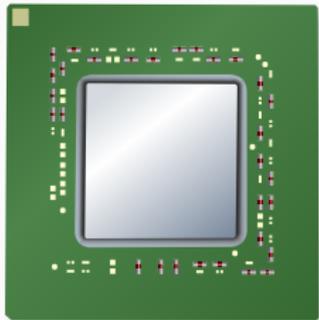


sys. d'exploitation

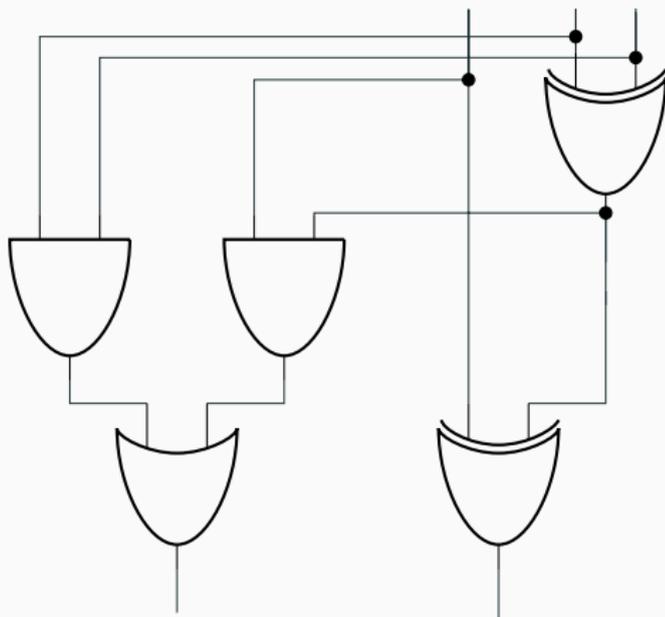


processeur





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!

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



© Spacewar! (1962)

```
tno, 6,      law i 41
tv1, 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)

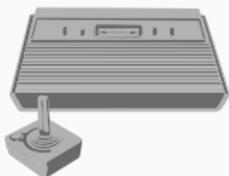


The screenshot shows the Droid Info application interface. At the top, there's a status bar with 'Koodo', signal strength, Wi-Fi, 55% battery, and 14:47. Below is a blue header with the Android robot icon, 'Droid Info', and icons for a document and settings. A navigation bar contains 'Dispositif', 'Système', and 'Mémoire'. The 'Système' tab is active, showing a 'PROCESSEUR' section with the following details:

PROCESSEUR	
Architecture du CPU	ARMv8-A
Board	EML
Chipset	KIRIN970
Coeurs	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>invalide</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?

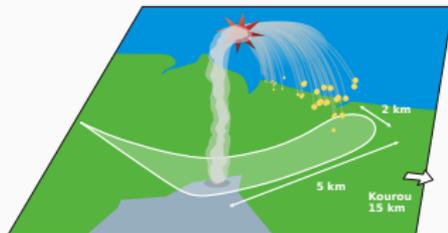
01000001011011000110110010010011

*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 jeudi!