

IFT209 – Programmation système
Université de Sherbrooke

Examen périodique

Enseignant: Michael Blondin
Date: jeudi 24 février 2022
Durée: 110 min.

Directives:

- Vous devez répondre aux questions dans le **cahier de réponses**, *pas* sur ce questionnaire;
- **Une seule feuille (recto verso)** de notes au format 8½" × 11" est permise;
- **Aucun matériel additionnel** (notes de cours, fiches récapitulatives, etc.) n'est permis;
- **Aucun appareil électronique** (calculatrice, téléphone, tablette, ordinateur, etc.) n'est permis;
- Vous devez donner **une seule réponse** par sous-question;
- L'examen comporte **6 questions** sur **5 pages** valant un total de **50 points**;
- La correction se base sur la **clarté**, l'**exactitude** et la **concision** de vos réponses, ainsi que sur la **justification** pour les questions qui en requièrent une;
- À moins d'avis contraire, le langage d'assemblage utilisé est celui de l'**architecture ARMv8** tel qu'utilisé en classe; un sommaire de cette architecture est présenté en **annexe**.

Question 1: systèmes de numération

(a)

2 pts

(b)

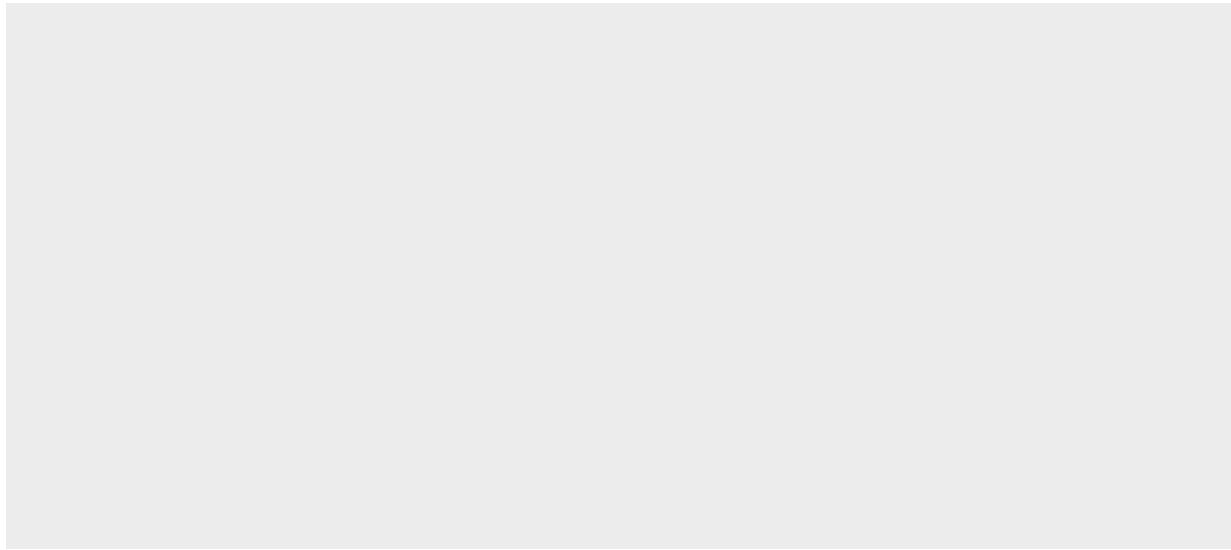
2 pts

(c)

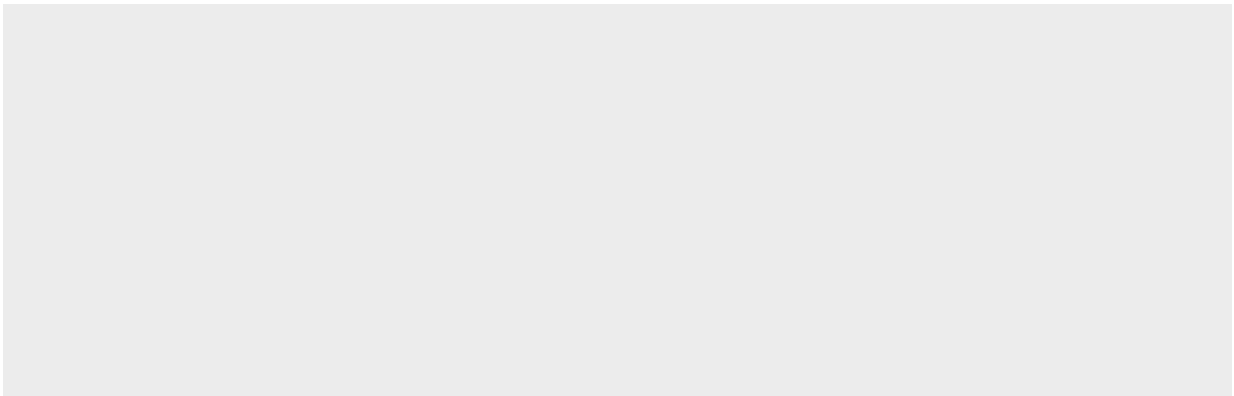
2 pts

(d)

2 pts

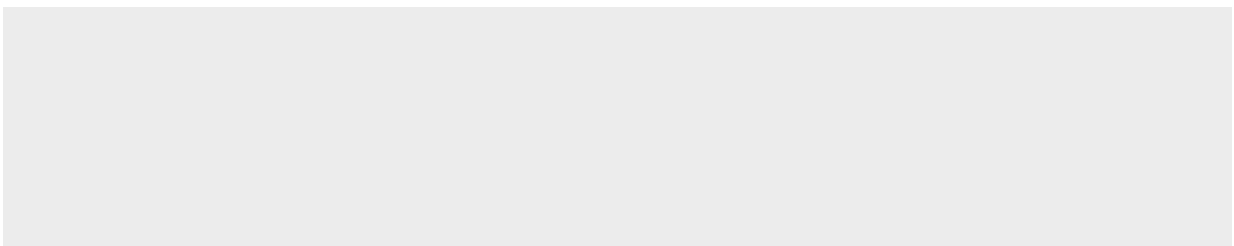
Question 2: architecture des ordinateurs

(a)



6 pts

(b)



2 pts

Question 3: mémoire et accès aux données

(a)

2 pts

(b)

2 pts

(c)

4 pts

Question 4: entiers signés et circuits logiques

(a)

3 pts

(b)

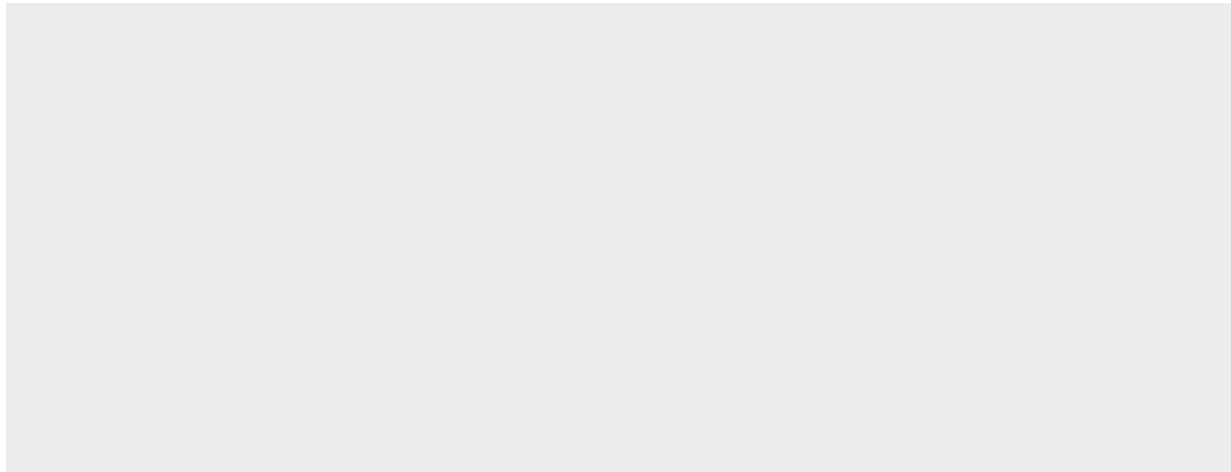
2 pts

(c)

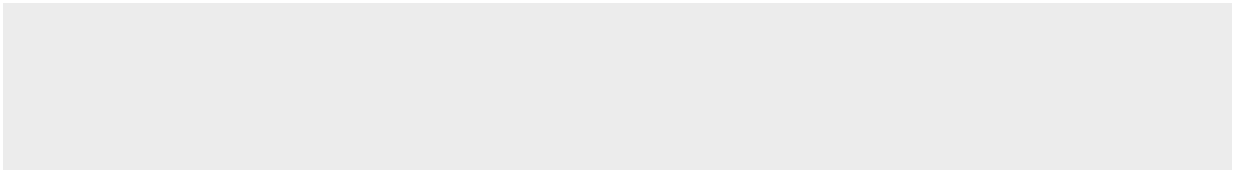
3 pts

Question 5: programmation en langage d'assemblage

8 pts

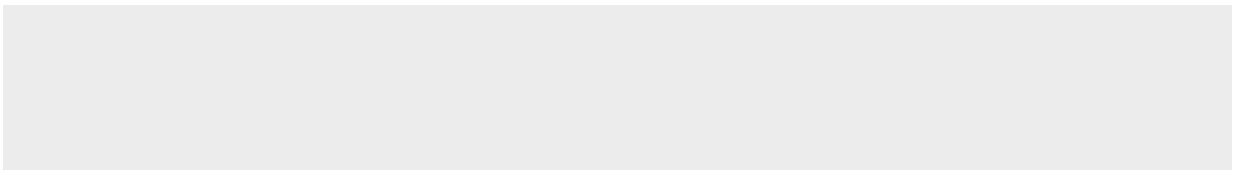
Question 6: tableaux

(a)



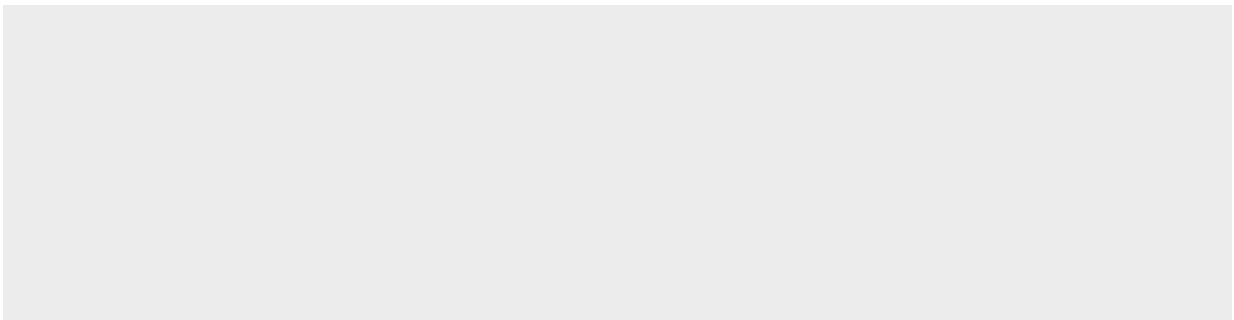
1,5 pts

(b)



1,5 pts

(c)



7 pts

Annexe:

Sommaire de l'architecture ARMv8

Registres.

- Chaque registre x_n possède 64 bits: $b_{63}b_{62} \dots b_1b_0$
- Notation: $x_n\langle i \rangle := b_i$, $x_n\langle i, j \rangle := b_i b_{i-1} \dots b_j$, r_n réfère au registre x_n ou w_n
- Chaque sous-registre w_n possède 32 bits et correspond à $x_n\langle 31, 0 \rangle$
- Le compteur d'instruction pc n'est pas accessible
- Conventions:

Registres	Nom	Utilisation
$x_0 - x_7$	—	registres d'arguments et de retour de sous-programmes
x_8	xr	registre pour retourner l'adresse d'une structure
$x_9 - x_{15}$	—	registres temporaires sauvegardés par l'appelant
$x_{16} - x_{17}$	ip ₀ - ip ₁	registres temporaires intra-procéduraux
x_{18}	pr	registre temporaire pouvant être réservé par le système
$x_{19} - x_{28}$	—	registres temporaires sauvegardés par l'appelé
x_{29}	fp	pointeur vers l'ancien sommet de pile (<i>frame pointer</i>)
x_{30}	lr	registre d'adresse de retour (<i>link register</i>)
x_{31}	sp	registre contenant la valeur 0, ou pointeur de pile (<i>stack pointer</i>)

Arithmétique (entiers).

- Les codes de condition sont modifiés par **cmp**, **adds**, **adcs**, **subs**, **sbc** et **negs**
- À cette différence près, **adds**, **adcs**, **subs**, **sbc** et **negs** se comportent respectivement comme **add**, **adc**, **sub**, **sbc** et **neg**
- Instructions, où i est une valeur immédiate de 12 bits et j est une valeur immédiate de 6 bits:

Code d'op.	Syntaxe	Effet	Exemple
cmp	cmp rd, rm	compare r_d et r_m	cmp x19, x21
	cmp rd, i	compare r_d et i	cmp x19, 42
	cmp rd, rm, decal j	compare r_d et r_m <i>decal j</i>	cmp x19, x21, lsl 1
add	add rd, rn, rm	$r_d \leftarrow r_n + r_m$	add x19, x20, x21
	add rd, rn, i	$r_d \leftarrow r_n + i$	add x19, x20, 42
	add rd, rn, rm, decal j	$r_d \leftarrow r_n + (r_m \text{ decal } j)$	add x19, x20, x21, lsl 1
adc	adc rd, rn, rm	$r_d \leftarrow r_n + r_m + C$	adc x19, x20, x21
sub	sub rd, rn, rm	$r_d \leftarrow r_n - r_m$	sub x19, x20, x21
	sub rd, rn, i	$r_d \leftarrow r_n - i$	sub x19, x20, 42
	sub rd, rn, rm, decal j	$r_d \leftarrow r_n - (r_m \text{ decal } j)$	sub x19, x20, x21, lsl 1
sbc	sbc rd, rn, rm	$r_d \leftarrow r_n - r_m - 1 + C$	sbc x19, x20, x21
neg	neg rd, rm	$r_d \leftarrow -r_m$	neg x19, x21
	neg rd, rm, decal j	$r_d \leftarrow -(r_m \text{ decal } j)$	neg x19, x21, lsl 1
mul	mul rd, rn, rm	$r_d \leftarrow r_n \cdot r_m$	mul x19, x20, x21
udiv	udiv rd, rn, rm	$r_d \leftarrow r_n \div r_m$ (non signé)	udiv x19, x20, x21
sdiv	sdiv rd, rn, rm	$r_d \leftarrow r_n \div r_m$ (signé)	sdiv x19, x20, x21
madd	madd rd, rn, rm, ra	$r_d \leftarrow r_a + (r_n \cdot r_m)$	madd x19, x20, x21, x22
msub	msub rd, rn, rm, ra	$r_d \leftarrow r_a - (r_n \cdot r_m)$	msub x19, x20, x21, x22

Accès mémoire.

- **ldrsb**, **ldrsh** et **ldrsb** se comportent respectivement comme **ldr** (4 octets), **ldrh** et **ldrb** à l'exception du fait qu'ils effectuent un chargement dans x_d où les bits excédentaires sont le bit de signe de la donnée chargée, plutôt que des zéros
- Instructions, où a est une adresse et $mem_b[a]$ réfère aux b octets à l'adresse a de la mémoire principale:

Code d'op.	Syntaxe	Effet	Exemple
mov	mov rd, rm mov rd, i	$r_d \leftarrow r_m$ $r_d \leftarrow i$	mov x19, x21 mov x19, 42
ldr	ldr xd, a ldr wd, a	charge 8 octets: $x_d \langle 63, 0 \rangle \leftarrow mem_8[a]$ charge 4 octets: $x_d \langle 31, 0 \rangle \leftarrow mem_4[a]$; $x_d \langle 63, 32 \rangle \leftarrow 0$	ldr x19, [x20] ldr w19, [x20]
ldrh	ldrh wd, a	charge 2 octets: $x_d \langle 15, 0 \rangle \leftarrow mem_2[a]$; $x_d \langle 63, 16 \rangle \leftarrow 0$	ldrh w19, [x20]
ldrb	ldrb wd, a	charge 1 octet: $x_d \langle 7, 0 \rangle \leftarrow mem_1[a]$; $x_d \langle 63, 8 \rangle \leftarrow 0$	ldrb w19, [x20]
str	str xd, a str wd, a	stocke 8 octets: $mem_8[a] \leftarrow x_d \langle 63, 0 \rangle$ stocke 4 octets: $mem_4[a] \leftarrow x_d \langle 31, 0 \rangle$	str x19, [x20] str w19, [x20]
strh	strh wd, a	stocke 2 octets: $mem_2[a] \leftarrow x_d \langle 15, 0 \rangle$	strh w19, [x20]
strb	strb wd, a	stocke 1 octet: $mem_1[a] \leftarrow x_d \langle 7, 0 \rangle$	strb w19, [x20]
ldp	ldp xd, xn, a	charge 16 octets: $x_d \langle 63, 0 \rangle \leftarrow mem_8[a]$, $x_n \langle 63, 0 \rangle \leftarrow mem_8[a+8]$	ldp x19, x20, [sp]
stp	stp xd, xn, a	stocke 16 octets: $mem_8[a] \leftarrow x_d \langle 63, 0 \rangle$, $mem_8[a+8] \leftarrow x_n \langle 63, 0 \rangle$	stp x19, x20, [sp]

Conditions de branchement.

- Codes de condition: N (négatif), Z (zéro), C (report), V (débordement)
- C indique aussi l'absence d'emprunt lors d'une soustraction
- Conditions de branchement:

Entiers non signés

Code	Signification	Codes de condition
eq	=	Z
ne	$\neq$	$\neg Z$
hs	$\geq$	C
hi	>	$C \wedge \neg Z$
ls	$\leq$	$\neg C \vee Z$
lo	<	$\neg C$

Entiers signés

Code	Signification	Codes de condition
eq	=	Z
ne	$\neq$	$\neg Z$
ge	$\geq$	$N = V$
gt	>	$\neg Z \wedge (N = V)$
le	$\leq$	$Z \vee (N \neq V)$
lt	<	$N \neq V$
vs	débordement	V
vc	pas de débordement	$\neg V$
mi	négatif	N
pl	non négatif	$\neg N$

Branchement.

- Instructions de branchement, où j est une valeur immédiate de 6 bits:

Code d'op.	Syntaxe	Effet	Exemple
b.	b.cond etiq	branche à etiq : si <i>cond</i>	b.eq main100
b	b etiq	branche à etiq :	b main100
cbz	cbz rd, etiq	branche à etiq : si $r_d = 0$	cbz x19, main100
cbnz	cbnz rd, etiq	branche à etiq : si $r_d \neq 0$	cbnz x19, main100
tbz	tbz rd, j, etiq	branche à etiq : si $r_d \langle j \rangle = 0$	tbz x19, 1, main100
tbnz	tbnz rd, j, etiq	branche à etiq : si $r_d \langle j \rangle \neq 0$	tbnz x19, 1, main100
bl	bl etiq	branche à etiq : et $x_{30} \leftarrow pc + 4$	bl printf
blr	blr xd	branche à x_d et $x_{30} \leftarrow pc + 4$	blr x20
br	br xd	branche à x_d	br x20
ret	ret	branche à x_{30} (retour de sous-prog.)	ret

Adressage.

► Modes d'adressages, où k est une valeur immédiate de 7 bits:

Nom	Syntaxe	Adresse	Effet	Exemple
adresse d'une étiquette	adr xd, etiq	—	$x_d \leftarrow$ adresse de etiq :	adr x19, main100
indirect par registre	[xd]	x_d	—	[x20]
indirect par registre indexé	[xd, xn]	$x_d + x_n$	—	[x20, x21]
	[xd, k]	$x_d + k$	—	[x20, 1]
	[xd, xn, decal k]	$x_d + (x_n \text{ decal } k)$	—	[x20, x21, lsl 1]
ind. par reg. indexé pré-inc.	[xd, k]!	$x_d + k$	$x_d \leftarrow x_d + k$ avant calcul	[x20, 1]!
ind. par reg. indexé post-inc.	[xd], k	x_d	$x_d \leftarrow x_d + k$ après calcul	[x20], 1
relatif	etiq	adresse de etiq	—	main100

Autres instructions.

Code d'op.	Syntaxe	Effet	Exemple
csel	csel rd, rn, rm, cond	si <i>cond</i> : $r_d \leftarrow r_n$, sinon: $r_d \leftarrow r_m$	csel x19, x20, x21, eq

Les manipulations de bits ne seront couvertes qu'après la relâche.

Logique et manipulation de bits.

- Les instructions **lsl**, **lsr**, **asr** et **ror** possèdent également une variante de 32 bits utilisant les registres w_d , w_n et w_m (dans ce cas, les 32 bits de poids fort sont mis à 0)
- Instructions, où i est une valeur immédiate de 12 bits et j est une valeur immédiate de 6 bits:

Code d'op.	Syntaxe	Effet	Exemple
mvn	mvn rd, rn	$r_d \leftarrow \neg r_n$	mvn x19, x20
and	and rd, rn, rm	$r_d \leftarrow r_n \wedge r_m$	and x19, x20, x21
	and rd, rn, i	$r_d \leftarrow r_n \wedge i$	and x19, x20, 4
	and rd, rn, rm, decal j	$r_d \leftarrow r_n \wedge (r_m \text{ decal } j)$	and x19, x20, x21, lsl 1
orr	orr rd, rn, rm	$r_d \leftarrow r_n \vee r_m$	orr x19, x20, x21
	orr rd, rn, i	$r_d \leftarrow r_n \vee i$	orr x19, x20, 4
	orr rd, rn, rm, decal j	$r_d \leftarrow r_n \vee (r_m \text{ decal } j)$	orr x19, x20, x21, lsl 1
eor	eor rd, rn, rm	$r_d \leftarrow r_n \oplus r_m$	eor x19, x20, x21
	eor rd, rn, i	$r_d \leftarrow r_n \oplus i$	eor x19, x20, 4
	eor rd, rn, rm, decal j	$r_d \leftarrow r_n \oplus (r_m \text{ decal } j)$	eor x19, x20, x21, lsl 1
bic	bic rd, rn, rm	$r_d \leftarrow r_n \wedge \neg r_m$	bic x19, x20, x21
	bic rd, rn, i	$r_d \leftarrow r_n \wedge \neg i$	bic x19, x20, 4
	bic rd, rn, rm, decal j	$r_d \leftarrow r_n \wedge \neg (r_m \text{ decal } j)$	bic x19, x20, x21, lsl 1
lsl	lsl xd, xn, j	décalage de j bits vers la gauche: $x_d \langle 63, j \rangle \leftarrow x_n \langle 63 - j, 0 \rangle$; $x_d \langle j - 1, 0 \rangle \leftarrow 0$	lsl x19, x20, 1
lsr	lsr xd, xn, j	décalage de j bits vers la droite: $x_d \langle 63 - j, 0 \rangle \leftarrow x_n \langle 63, j \rangle$; $x_d \langle 63, 64 - j \rangle \leftarrow 0$	lsr x19, x20, 1
asr	asr xd, xn, j	décalage arithmétique de j bits vers la droite: $x_d \langle 63 - j, 0 \rangle \leftarrow x_n \langle 63, j \rangle$; $x_d \langle 63, 64 - j \rangle \leftarrow x_n \langle 63 \rangle$	asr x19, x20, 1
ror	ror xd, xn, j	décalage circulaire de j bits vers la droite: $x_d \leftarrow x_n \langle j - 1, 0 \rangle x_n \langle 63, j \rangle$	ror x19, xn, 1

Données statiques.

Segments de données		Données	
Pseudo-instruction	Contenu		
<code>.section ".text"</code>	instructions	<code>.align</code> k	donnée suivante stockée à une adresse divisible par k
<code>.section ".rodata"</code>	données en lecture seule	<code>.skip</code> k	réserve k octets
<code>.section ".data"</code>	données initialisées	<code>.ascii</code> s	chaîne de caractères initialisée à s
<code>.section ".bss"</code>	données non-initialisées	<code>.asciz</code> s	chaîne de caractères initialisée à s suivi du carac. nul
		<code>.byte</code> v	octet initialisé à v
		<code>.hword</code> v	demi-mot initialisé à v
		<code>.word</code> v	mot initialisé à v
		<code>.xword</code> v	double mot initialisé à v
		<code>.single</code> f	nombre en virg. flottante simple précision initialisé à f
		<code>.double</code> f	nombre en virg. flottante double précision initialisé à f

Entrées/sorties (haut niveau).

- Affichage: `printf(&format, val1, val2, ...)`
- Lecture: `scanf(&format, &var1, &var2, ...)`
- Spécificateurs de format:

Famille	Format	Type
Nombres sur 64 bits	<code>%ld</code>	entier décimal signé
	<code>%lu</code>	entier décimal non signé
	<code>%lX</code>	entier hexadécimal non signé
	<code>%lf</code>	nombre en virgule flottante
Nombres sur 32 bits	<code>%d</code>	entier décimal signé
	<code>%u</code>	entier décimal non signé
	<code>%X</code>	entier hexadécimal non signé
	<code>%f</code>	nombre en virgule flottante
Nombres sur 16 bits	<code>%hd</code>	entier décimal signé
	<code>%hu</code>	entier décimal non signé
	<code>%hX</code>	entier hexadécimal non signé
	<code>%c</code>	caractère (1 octet)
Caractères	<code>%s</code>	chaîne de caractères