INF2171 — Organisation des ordinateurs et assembleur

Examen Final — Automne 2023

Jean Privat — Dimanche 17 décembre — Durée 3 heures

- *Aucun document n'est autorisé sauf une feuille de note manuscrite au format lettre (recto-verso).*
- *L'usage de la calculatrice ou de tout appareil électronique est interdit.*
- *Répondez et rendez seulement le formulaire (avant dernière feuille).*
- *L'annexe est détachable (dernière feuille).*
- *Les questions faisant apparaître le symbole ♣ peuvent présenter plusieurs bonnes réponses. Les autres ont une unique bonne réponse.*
- *Chacune des 20 questions vaut 5 points. Des points négatifs pourront être affectés à de* très mauvaises *réponses.*

1 Structure et fonctionnement d'un ordinateur

Question 1 En RISC-V, comment est sauvegardé l'adresse de retour lors d'un appel de routine?

- [A] Elle est écrite dans un registre par `jal`.
- [B] Elle est gérée manuellement par le programmeur.
- [C] Elle est empilée dans la pile par `bgt`.
- [D] Elle est empilée dans la pile par `ret`.
- [E] Elle est empilée dans la pile par `jal`.
- [F] Elle est écrite dans un registre par `ret`.
- [G] Elle est écrite dans un registre par `bgt`.
- [H] Il n'y a pas de retour possible.

Question 2 ♣ Parmi les propositions suivantes à propos de la mémoire, en particulier la pile (*stack*) et le tas (*heap*), indiquez toutes celles qui sont vraies.

- [A] Le pointeur de pile est stocké dans un registre du micro-processeur.
- [B] Le code machine des routines est alloué dans la pile.
- [C] Les données dynamiquement allouées avec `Sbrk` sont allouées dans le tas.
- [D] Le tas a une taille infinie.
- [E] Allouer de la pile se fait en décrémentant le pointeur de pile.
- [F] Les variables et données globales sont allouées dans la pile.
- [G] Les variables locales d'une routine peuvent êtres allouées dans la pile.
- [H] Avant d'effectuer des calculs intensifs, il est recommandé de télécharger plus de RAM.

Question 3 ♣ Que doit faire l'unité centrale de traitement lors d'une interruption logicielle (ou *exception* en RISC-V) ? Indiquez toutes les bonnes réponses.

- [A] Armer une IRQ
- [B] Attendre une IRQ
- [C] Brancher sur la gérante d'interruption
- [D] Sortir récursivement de toutes les routines
- [E] Sauvegarder le compteur ordinal
- [F] Désarmer une IRQ
- [G] Afficher un message d'erreur
- [H] Réinitialiser le compteur ordinal à 0

brouillon

2 Programme mystère

Soit le programme mystérieux suivant:

```
        li a7, 1
        li a0, 2
        jal jal
        li a0, 3
jal:    ecall
        li a0, 4
        jalr ra
        li a0, 5
        ecall
        li a0, 6
        jalr ra
        li a0, 7
        ecall
        li a7, 10
        ret
```

Question 4 Quel est le premier chiffre affiché par ce programme mystérieux ?

☐ A 3	☐ C 0	☐ E 7	☐ G 5	☐ I *Aucune de ces réponses*
☐ B 4	■ D 2	☐ F 6	☐ H 1	

Question 5 Quel est le second chiffre affiché par ce programme mystérieux ?

☐ A 7	☐ C 6	☐ E 2	☐ G 0	☐ I *Aucune de ces réponses*
■ B 3	☐ D 4	☐ F 1	☐ H 5	

Question 6 Quel est le troisième chiffre affiché par ce programme mystérieux ?

■ A 5	☐ C 6	☐ E 7	☐ G 4	☐ I *Aucune de ces réponses*
☐ B 0	☐ D 3	☐ F 2	☐ H 1	

Question 7 Quel est le quatrième chiffre affiché par ce programme mystérieux ?

■ A 5	☐ C 0	☐ E 4	☐ G 6	☐ I *Aucune de ces réponses*
☐ B 3	☐ D 1	☐ F 2	☐ H 7	

3 Routines et matrices

Soit le programme incomplet suivant qui doit afficher un caractère d'une matrice en fonction de ses coordonnées. Une fois complété, si l'on saisi « `3 0` », le programme affiche `X` (celui de la ligne du haut). Et si l'on saisi « `4 1` », il affiche `O` (celui de la colonne de droite).

```
        .data
        .eqv matW, 5    # largeur de mat
        .eqv matH, 3    # hauteur de mat
mat:    .ascii "   X "
        .ascii " XXOO"
        .ascii "XOXOX"
        # Appels système RARS utilisés
        .eqv ReadInt, 5
        .eqv Exit, 10
        .eqv PrintChar, 11
        .text
        # Lecture des coordonnées
        li a7, ReadInt
        ecall
        mv s0, a0       # colonne (x)
        ecall
        mv s1, a0       # ligne (y)
        # Appel de get(x,y,mat)
        mv a0, s0
        mv a1, s1
        # TODO 1 ******
        # Affichage du caractère et terminaison
        li a7, PrintChar
        ecall
        li a7, Exit
        ecall
# get: retourne un caractère de la matrice
# * a0: colonne dans la matrice (x)
# * a1: ligne dans la matrice (y)
# * a2: adresse de la matrice
# * retour a0: valeur du caractère (-1 si en dehors)
# * retour a1: adresse du caractère (-1 si en dehors)
get:    # Test des bornes
        bltz a0, getbad
        bltz a1, getbad
        li t0, matW
        li t1, matH
        # TODO 2 ******
        # Calcul dans a1 de l'adresse du caractère
        # TODO 3 ******
        add a1, a1, a2
        # Récupération dans a0 du caractère
        lbu a0, 0(a1)
        j getret
getbad: li a0, -1
        li a1, -1
getret: ret
```

Question 8 Que doit-on mettre pour finir l'appel à la routine `get`, à la place de `TODO 1` ?

A
```
ld a2, mat
jalr get
```
B
```
lb a2, mat
j get
```
C
```
li a2, mat
j get
```
D
```
la a2, mat
jal get
```
E
```
ld a2, mat
j get
```
F
```
lb a2, mat
jal get
```
G
```
la a2, mat
jalr get
```
H
```
li a2, mat
jalr get
```
I
```
lb a2, mat
jalr get
```
J
```
li a2, mat
jal get
```
K
```
la a2, mat
j get
```
L
```
ld a2, mat
jal get
```

Question 9 Que doit-on mettre pour finir le test des bornes dans la routine `get`, à la place de `TODO 2` ?

A
```
blt a0, t0, getret
blt a1, t1, getret
```
B
```
blt a0, t0, get
blt a1, t1, get
```
C
```
bgt a0, t0, getret
bgt a1, t1, getret
```
D
```
bge a0, t0, getbad
bge a1, t1, getbad
```
E
```
bge a0, t0, getret
bge a1, t1, getret
```
F
```
blt a0, t0, getbad
blt a1, t1, getbad
```
G
```
ble a0, t0, getbad
ble a1, t1, getbad
```
H
```
ble a0, t0, getret
ble a1, t1, getret
```
I
```
bgt a0, t0, getbad
bgt a1, t1, getbad
```

Question 10 Que doit-on mettre pour compléter le calcul de l'adresse du caractère dans la routine `get`, à la place de `TODO 3` ?

A
```
mul a1, a1, a0
sll a1, a1, t1
```
B
```
add a1, a1, a0
mul a1, a1, t0
```
C
```
add a1, a1, a0
mul a1, a1, t1
```
D
```
mul a1, a1, t1
add a1, a1, a0
```
E
```
sll a1, a1, t0
add a1, a1, a0
```
F
```
sll a1, a1, t1
add a1, a1, a0
```
G
```
add a1, a1, a0
sll a1, a1, t1
```
H
```
add a1, a1, a0
sll a1, a1, t0
```
I
```
mul a1, a1, t0
add a1, a1, a0
```

brouillon

4 Allocation dans la pile et récursivité

Soit la routine `ack` incomplète suivante:

```
ack:
        # Prologue (à terminer)
        addi    sp, sp, ???
        # [...]
        # Corps de la routine
        mv      s0, a0
        blez    a0, ackfin
ackloop:
        blez    a1, ackmid
        addi    a1, a1, -1
        mv      a0, s0
        jal     ack
        mv      a1, a0
        j       ackcont
ackmid:
        li      a1, 1
ackcont:
        addi    s0, s0, -1
        bgtz    s0, ackloop
ackfin:
        addi    a0, a1, 1
        # Épilogue (à écrire)
        # [...]
        ret
```

Question 11 Quelle valeur devrait-être utilisée pour remplacer `???` dans l'instruction « `addi sp, sp, ???` » ?

A `-32`	C `-4`	E `128`	G `8`	I `-64`	K `-128`	M `32`	O `-16`
B `2`	D `4`	F `-8`	H `1`	J `64`	L `16`	N `-1`	P `-2`

Question 12 Quelle instruction devrait apparaitre parmi celles de l'épilogue ?

A `lw sp, 8(s0)`	E `lw sp, 8(a0)`	I `ld sp, 8(s0)`	M `sw sp, 8(a0)`
B `lw s0, 8(sp)`	F `sd s0, 8(sp)`	J `sd a0, 8(sp)`	N `sd sp, 8(a0)`
C `lw a0, 8(sp)`	G `sw sp, 8(s0)`	K `sd sp, 8(s0)`	O `ld a0, 8(sp)`
D `ld s0, 8(sp)`	H `sw a0, 8(sp)`	L `ld sp, 8(a0)`	P `sw s0, 8(sp)`

Question 13 Avec les arguments `a0`=2 et `a1`=1, la routine `ack` retourne `a0`=5.
Afin, de déterminer la pile consommé, indiquez combien de cadres d'exécution (*stack frames*) de `ack` sont empilés à la fois au maximum pour `ack(2,1)` (on parle aussi de profondeur de récursion) ?
Note: faites cette question en dernier.

A 6	D 5	G 64	J 8
B 4	E 1	H 0	K 16
C 3	F 2	I 7	L 32

brouillon

5 Adressage dans le tas

Soit le programme suivant. On considère que le tas de RARS commence à l'adresse `0x10200000` et que les allocations sont alignées à 8 octets.

```
# Structure avec 3 champs
.eqv l, 0
.eqv r, 8
.eqv d, 16
.eqv s, 24
# Appel système RARS utilisé
.eqv Sbrk, 9

# Un
li a7, Sbrk
li a0, s
ecall
mv s0, a0
li a0, 8
sd a0, d(s0)

# Deux
li a7, Sbrk
li a0, s
ecall
mv s1, a0
sd s0, l(s1)
sd s1, r(s0)
li a0, 16
sd a0, d(s1)
```

```
# Trois
li a7, Sbrk
li a0, s
ecall
mv s2, a0
sd s1, l(s2)
sd s0, r(s2)
sd s2, l(s0)
sd s2, r(s1)
li a0, 24
sd a0, d(s2)

# a1
mv a1, s1
# a2
ld t0, l(s1)
ld a2, d(t0)
# a3
ld t1, l(s1)
ld t1, l(t1)
ld t1, r(t1)
ld t1, r(t1)
ld t1, l(t1)
ld t1, l(t1)
ld t1, l(t1)
ld a3, d(t1)

ebreak
```

Question 14 Au moment du `ebreak`, quel est l'octet de poids faible du registre `a1` ?

[A] 0x00	[C] 0x10	[E] 0x20	[G] 0x30	[I] 0x40	[K] 0xD0	[M] 0xE0	[O] 0xF0
[B] 0x08	[D] 0x18	[F] 0x28	[H] 0x38	[J] 0x48	[L] 0xD8	[N] 0xE8	[P] 0xF8

Question 15 Même question pour le registre `a2` ?

[A] 0x00	[C] 0x10	[E] 0x20	[G] 0x30	[I] 0x40	[K] 0xD0	[M] 0xE0	[O] 0xF0
[B] 0x08	[D] 0x18	[F] 0x28	[H] 0x38	[J] 0x48	[L] 0xD8	[N] 0xE8	[P] 0xF8

Question 16 Même question pour le registre `a3` ?

[A] 0x00	[C] 0x10	[E] 0x20	[G] 0x30	[I] 0x40	[K] 0xD0	[M] 0xE0	[O] 0xF0
[B] 0x08	[D] 0x18	[F] 0x28	[H] 0x38	[J] 0x48	[L] 0xD8	[N] 0xE8	[P] 0xF8

brouillon

6 Nombres flottants

Rappels. IEEE 754 simple-précision (binary32) : 1 bit de signe, 8 bits d'exposant, pôle: 127 (`0x7F`), 23 bits de mantisse. IEEE 754 double-précision (binary64) : 1 bit de signe, 11 bits d'exposant, pôle: 1023 (`0x3FF`), 52 bits de mantisse.

Question 17 Laquelle des représentations binaires suivantes correspond au nombre $8.75_{(10)}$?

- [x] A $1.00011_{(2)} \times 2^3$
- [ ] B $1.0011_{(2)} \times 2^{-2}$
- [ ] C $1.0011_{(2)} \times 2^{-3}$
- [ ] D $1.00011_{(2)} \times 2^{-3}$
- [ ] E $1.0011_{(2)} \times 2^2$
- [ ] F $1.00011_{(2)} \times 2^{-2}$
- [ ] G $1.00011_{(2)} \times 2^2$
- [ ] H $1.0011_{(2)} \times 2^3$

Question 18 Lequel des flottants IEEE 745 simple-précision suivants représente le nombre -10.125054 ?

- [ ] A `0xC2520913`
- [ ] B `0xC1520123`
- [x] C `0xC1220039`
- [ ] D `0x4152FFFF`
- [ ] E `0x42520913`
- [ ] F `0x42229912`
- [ ] G `0xC2220913`
- [ ] H `0x41220139`

Question 19 ♣ Soit un nombre flottant IEEE 754 simple-précision codé en hexadécimal par `0x42??????` (l'octet de poids fort est `0x42`, les autres octets sont inconnus). Quelles sont les valeurs possibles de ce nombre ?
Note: tous les nombres proposés sont exactement représentables en IEEE 754 simple-précision.

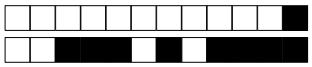
- [ ] A 256
- [x] B 127
- [x] C 127.99999237060546875
- [x] D 63.999996185302734375
- [ ] E 255
- [ ] F 31.9999980926513671875
- [x] G 64
- [x] H 63
- [ ] I 31
- [ ] J 255.9999847412109375
- [x] K 32
- [ ] L 128

Question 20 On souhaite concevoir un circuit logique combinatoire qui permet de calculer la racine carrée d'un nombre flottant IEEE 754 double-précision.
Les entrées du circuit sont: les 64 bits qui codent le nombre et 3 bits qui représentent l'arrondi demandé. Les sorties du circuit sont: les 64 bits qui codent le résultat et 4 bits de drapeaux qui codent une exception des IEEE 754 possible : opération invalide, débordement (*overflow*), débordement par dessous (*underflow*), inexactitude (seulement quatre exceptions car la division par zéro n'est pas possible ici).
Admettons que l'on souhaite écrire la table de vérité du circuit, combien de lignes seraient nécessaires ?

- [ ] A 2^{64+4}
- [ ] B $(64+3)^2$
- [ ] C $2^{64+3+64+4}$
- [ ] D $2 \times (64+4)$
- [ ] E $2 \times (64+3+64+4)$
- [x] F 2^{64+3}
- [ ] G $2 \times (64+3)$
- [ ] H $(64+3)^2$
- [ ] I $64+4$
- [ ] J $(64+3+64+4)^2$
- [ ] K $64+3$
- [ ] L $64+3+64+4$

brouillon

brouillon



+1/14/47+

Feuille de réponses, final automne 2023
INF2171 — Organisation des ordinateurs et assembleur

0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9

⟵ codez les 8 chiffres de votre code permanent ci-contre, et inscrivez-le à nouveau ci-dessous avec votre nom et prénom. 5 points de pénalité en cas d'oubli ou d'erreur.

Code permanent :

.....

Nom :

.....

Prénom :

.....

Les réponses aux questions sont à donner exclusivement sur cette feuille : les réponses données sur les feuilles précédentes ne seront pas prises en compte. Ne pas utiliser un formulaire d'une autre copie au risque d'avoir toutes les réponses fausses.
Important : noircissez complètement l'intérieur de chaque case (pas de croix, pas de cercle).

Question 1 : ■ B C D E F G H

Question 2 : ■ B ■ D ■ F ■ H

Question 3 : A B ■ D ■ F G H

Question 4 : A B C ■ E F G H I

Question 5 : A ■ C D E F G H I

Question 6 : ■ B C D E F G H I

Question 7 : ■ B C D E F G H I

Question 8 : A B C ■ E F G H I J K L

Question 9 : A B C ■ E F G H I

Question 10 : A B C D E F G H ■

Question 11 : A B C D E F G H I J K L M N ■ P

Question 12 : A B C ■ E F G H I J K L M N O P

Question 13 : A ■ C D E F G H I J K L

Question 14 : A B C ■ E F G H I J K L M N O P

Question 15 : A ■ C D E F G H I J K L M N O P

Question 16 : A B ■ D E F G H I J K L M N O P

Question 17 : ■ B C D E F G H

Question 18 : A B ■ D E F G H

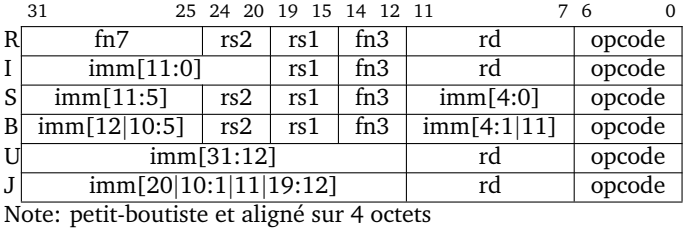
Question 19 : A ■ ■ D E F ■ ■ I J ■ L

Question 20 : A B C D E ■ G H I J K L

Jeu d'instruction RISC-V

Inst	Description	FMT	opcode	fn3	fn7 (ou imm)	Description	Note
add	Add	R	0x33	0x0	0x00	rd = rs1 + rs2	
sub	Substract	R	0x33	0x0	0x20	rd = rs1 - rs2	
xor	Exclusive Or	R	0x33	0x4	0x00	rd = rs1 ^ rs2	
or	Or	R	0x33	0x6	0x00	rd = rs1 | rs2	
and	And	R	0x33	0x7	0x00	rd = rs1 & rs2	
sll	Shift Left Logical	R	0x33	0x1	0x00	rd = rs1 << rs2	
srl	Shift Right Logical	R	0x33	0x5	0x00	rd = rs1 >> rs2	
sra	Shift Right Arithmetic	R	0x33	0x5	0x20	rd = rs1 >> rs2	
slt	Set Less Than	R	0x33	0x2	0x00	rd = (rs1 < rs2)?1:0	
sltu	Set Less Than (U)	R	0x33	0x3	0x00	rd = (rs1 < rs2)?1:0	
addi	Add Immediate	I	0x13	0x0		rd = rs1 + imm	
xori	Xor Immediate	I	0x13	0x4		rd = rs1 ^ imm	
ori	Or Immediate	I	0x13	0x6		rd = rs1 | imm	
andi	And Immediate	I	0x13	0x7		rd = rs1 & imm	
slli	Shift Left Logical Imm	I	0x13	0x1	imm[11:6]=0x00	rd = rs1 << imm[5:0]	
srli	Shift Right Logical Imm	I	0x13	0x5	imm[11:6]=0x00	rd = rs1 >> imm[5:0]	
srai	Shift Right Arith Imm	I	0x13	0x5	imm[11:6]=0x10	rd = rs1 >> imm[5:0]	
slti	Set Less Than Imm	I	0x13	0x2		rd = (rs1 < imm)?1:0	
sltiu	Set Less Than Imm (U)	I	0x13	0x3		rd = (rs1 < imm)?1:0	
lb	Load Byte	I	0x03	0x0		rd = M[rs1+imm][7:0]	
lh	Load Half	I	0x03	0x1		rd = M[rs1+imm][15:0]	
lw	Load Word	I	0x03	0x2		rd = M[rs1+imm][31:0]	
ld	Load Double Word	I	0x03	0x3		rd = M[rs1+imm][63:0]	RV64I
lbu	Load Byte (U)	I	0x03	0x4		rd = M[rs1+imm][7:0]	
lhu	Load Half (U)	I	0x03	0x5		rd = M[rs1+imm][15:0]	
lwu	Load Word (U)	I	0x03	0x6		rd = M[rs1+imm][63:0]	RV64I
sb	Store Byte	S	0x23	0x0		M[rs1+imm][7:0] = rs2[7:0]	
sh	Store Half	S	0x23	0x1		M[rs1+imm][15:0] = rs2[15:0]	
sw	Store Word	S	0x23	0x2		M[rs1+imm][31:0] = rs2[31:0]	
sd	Store Double Word	S	0x23	0x3		M[rs1+imm][63:0] = rs2[63:0]	RV64I
beq	Branch ==	B	0x63	0x0		if(rs1 == rs2) PC += imm	
bne	Branch !=	B	0x63	0x1		if(rs1 != rs2) PC += imm	
blt	Branch <	B	0x63	0x4		if(rs1 < rs2) PC += imm	
bge	Branch $\geq$	B	0x63	0x5		if(rs1 >= rs2) PC += imm	
bltu	Branch < (U)	B	0x63	0x6		if(rs1 < rs2) PC += imm	
bgeu	Branch $\geq$ (U)	B	0x63	0x7		if(rs1 >= rs2) PC += imm	
jal	Jump And Link	J	0x6F			rd = PC+4; PC += imm	
jalr	Jump And Link Register	I	0x67	0x0		rd = PC+4; PC = rs1 + imm	
lui	Load Upper Imm	U	0x37			rd = imm << 12	
auipc	Add Upper Imm to PC	U	0x17			rd = PC + (imm << 12)	
ecall	Environment Call	I	0x73	0x0	imm=0x0	Appel système	
ebreak	Environment Break	I	0x73	0x0	imm=0x1	Appel au débogueur	
mul	Multiply	R	0x33	0x0	0x01	rd = (rs1 * rs2)[31:0]	RV32M
mulh	Multiply High	R	0x33	0x1	0x01	rd = (rs1 * rs2)[63:32]	RV32M
mulsu	Multiply High (S) (U)	R	0x33	0x2	0x01	rd = (rs1 * rs2)[63:32]	RV32M
mulu	Multiply High (U)	R	0x33	0x3	0x01	rd = (rs1 * rs2)[63:32]	RV32M
div	Divide	R	0x33	0x4	0x01	rd = rs1 / rs2	RV32M
divu	Divide (U)	R	0x33	0x5	0x01	rd = rs1 / rs2	RV32M
rem	Remainder	R	0x33	0x6	0x01	rd = rs1 % rs2	RV32M
remu	Remainder (U)	R	0x33	0x7	0x01	rd = rs1 % rs2	RV32M

Format des instructions



Note: petit-boutiste et aligné sur 4 octets

Registres

Registre	Nom d'ABI	Description	Qui sauve ?
x0	zero	Constante zero	—
x1	ra	Return address	Appellant
x2	sp	Stack pointer	Appelé
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5-x7,x28-x31	t0-t6	Temporaires	Appellant
x8,x9,x18-x27	s0-s11	Sauvegardés	Appelé
x10-x17	a0-a7	Arguments (et retours)	Appellant

Pseudoinstructions

Pseudoinstruction	Instruction(s)	Description
`la rd, lab`	`auipc rd, H` `addi rd, rd, L]`	Load Address
`l[bhwd] rd, lab`	`auipc rd, H` `l[bhwd] rd, L(rd)`	Load
`s[bhwd] rd, lab, rt`	`auipc rt, H` `s[bhwd] rd, L(rt)`	Store
`nop`	`addi x0, x0, 0`	No Operation
`li rd, imm`	`addi rd, x0, imm`	Load Immediate
`li rd, imm`	`Myriade`	Load Immediate (v2)
`mv rd, rs`	`addi rd, rs, 0`	Move (copie)
`not rd, rs`	`xori rd, rs, -1`	Not (bit à bit)
`neg rd, rs`	`sub rd, x0, rs`	Negate (opposé)
`seqz rd, rs`	`sltiu rd, rs, 1`	Set if $=$ zero
`snez rd, rs`	`sltu rd, x0, rs`	Set if $\neq$ zero
`sltz rd, rs`	`slt rd, rs, x0`	Set if $<$ zero
`sgtz rd, rs`	`slt rd, x0, rs`	Set if $>$ zero
`beqz rs, lab`	`beq rs, x0, lab`	Branch if $=$ zero
`bnez rs, lab`	`bne rs, x0, lab`	Branch if $\neq$ zero
`blez rs, lab`	`bge x0, rs, lab`	Branch if $\leq$ zero
`bgez rs, lab`	`bge rs, x0, lab`	Branch if $\geq$ zero
`bltz rs, lab`	`blt rs, x0, lab`	Branch if $<$ zero
`bgtz rs, lab`	`blt x0, rs, lab`	Branch if $>$ zero
`bgt rs, rt, lab`	`blt rt, rs, lab`	Branch if $>$
`ble rs, rt, lab`	`bge rt, rs, lab`	Branch if $\leq$
`bgtu rs, rt, lab`	`bltu rt, rs, lab`	Branch if $>$ (U)
`bleu rs, rt, lab`	`bgeu rt, rs, lab`	Branch if $\leq$ (U)
`j lab`	`jal x0, lab`	Jump
`jal lab`	`jal ra, lab`	Jump And Link
`jr rs`	`jalr x0, rs, 0`	Jump Register
`jalr rs`	`jalr ra, rs, 0`	JAL Register
`ret`	`jalr x0, ra, 0`	Return
`call lab`	`auipc x1, H` `jalr x1, x1, L`	Call (loin)
`tail lab`	`auipc x6, H` `jalr x0, x6, L`	Tail call (loin)
`fence`	`fence iorw, iorw`	Fence (totale)

Directives RARS et GNU as

Directive	Signification
`.align` n	Des octets à 0 pour aligner sur 2^n
`.ascii` s	Code ascii de chacun des caractères de s
`.asciz` s	Code ascii de chacun des caractères de s suivi de `'\0'`
`.byte` n	Une ou plusieurs valeurs sur un octet
`.data`	Travaille dans la section Data
`.double` n	Une ou plusieurs valeurs flotantes double précision
`.dword` n	Une ou plusieurs valeurs sur 8 octets (double mot)
`.eqv` s, n	Attribue la valeur n au symbole s
`.float` n	Une ou plusieurs valeurs flotantes simple précision
`.half` n	Une ou plusieurs valeurs sur 2 octets (demi mot)
`.space` n	n octets à 0
`.string` s	Alias pour .asciz
`.text`	Travaille dans la section Text
`.word` n	Une ou plusieurs valeurs sur 4 octets (mot)

Quelques appels système RARS

Nom	a7	Signification
PrintInt	1	Afficher le nombre `a0` (décimal)
PrintString	4	Afficher la chaîne pointée par `a0`
ReadInt	5	Lire nombre dans `a0` (décimal)
ReadString	8	Lire chaine dans le tampon `a0` de taille `a1`
Sbrk	9	Alloue `a0` octets dans le tas, retourne l'adresse dans `a0`
Exit	10	Quitter
PrintChar	11	Afficher le caractère de code ASCII `a0`
ReadChar	12	Saisir un caractère dans `a0` (code ASCII)

Codes ASCII importants

Hex	Dec	Caractère
`0x00`	0	NUL (fin de chaîne)
`0x0A`	10	Saut de ligne `'\n'`
`0x20`	32	Espace `' '`
`0x2A`	42	Étoile `'*'`
`0x30`	48	Premier chiffre `'0'`
`0x41`	65	Première majuscule `'A'`
`0x61`	97	Première minuscule `'a'`

Taille des entiers

Bits	Octets	Nom RISC-V	Signé	Min	Max
8	1	byte	non	0	255
			oui	-128	127
16	2	halfword	non	0	$65\,535$
			oui	$-32\,768$	$32\,767$
32	4	word	non	0	$\approx 4.29 \times 10^9$
			oui	$\approx -2.15 \times 10^9$	$\approx 2.15 \times 10^9$
64	8	doubleword	non	0	$\approx 1.84 \times 10^{19}$
			oui	$\approx -9.22 \times 10^{18}$	$\approx 9.22 \times 10^{18}$
128	16	quadword	non	0	$\approx 3.40 \times 10^{38}$
			oui	$\approx -1.70 \times 10^{38}$	$\approx 1.70 \times 10^{38}$