

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?

- | | |
|--|--|
| <input type="checkbox"/> A Elle est écrite dans un registre par <code>jal</code> . | <input type="checkbox"/> E Elle est empilée dans la pile par <code>jal</code> . |
| <input type="checkbox"/> B Elle est gérée manuellement par le programmeur. | <input type="checkbox"/> F Elle est écrite dans un registre par <code>ret</code> . |
| <input type="checkbox"/> C Elle est empilée dans la pile par <code>bgt</code> . | <input type="checkbox"/> G Elle est écrite dans un registre par <code>bgt</code> . |
| <input type="checkbox"/> D Elle est empilée dans la pile par <code>ret</code> . | <input type="checkbox"/> 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 être 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.

- | | |
|--|--|
| <input type="checkbox"/> A Armer une IRQ | <input type="checkbox"/> E Sauvegarder le compteur ordinal |
| <input type="checkbox"/> B Attendre une IRQ | <input type="checkbox"/> F Désarmer une IRQ |
| <input type="checkbox"/> C Brancher sur la gérante d'interruption | <input type="checkbox"/> G Afficher un message d'erreur |
| <input type="checkbox"/> D Sortir récursivement de toutes les routines | <input type="checkbox"/> 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 0 (celui de la colonne de droite).

```

.data
.eqv matW, 5      # largeur de mat
.eqv matH, 3      # hauteur de mat
mat:
.ascii "  X "
.ascii " XX00"
.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` ?

- | | | | |
|---|--|---|--|
| <input type="checkbox"/> A <code>ld a2, mat
jalr get</code> | <input type="checkbox"/> D <code>la a2, mat
jal get</code> | <input type="checkbox"/> G <code>la a2, mat
jalr get</code> | <input type="checkbox"/> J <code>li a2, mat
jal get</code> |
| <input type="checkbox"/> B <code>lb a2, mat
j get</code> | <input type="checkbox"/> E <code>ld a2, mat
j get</code> | <input type="checkbox"/> H <code>li a2, mat
jalr get</code> | <input type="checkbox"/> K <code>la a2, mat
j get</code> |
| <input type="checkbox"/> C <code>li a2, mat
j get</code> | <input type="checkbox"/> F <code>lb a2, mat
jal get</code> | <input type="checkbox"/> I <code>lb a2, mat
jalr get</code> | <input type="checkbox"/> L <code>ld a2, mat
jal get</code> |

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

- | | | |
|---|---|---|
| <input type="checkbox"/> A <code>blt a0, t0, getret
blt a1, t1, getret</code> | <input type="checkbox"/> D <code>bge a0, t0, getbad
bge a1, t1, getbad</code> | <input type="checkbox"/> G <code>ble a0, t0, getbad
ble a1, t1, getbad</code> |
| <input type="checkbox"/> B <code>blt a0, t0, get
blt a1, t1, get</code> | <input type="checkbox"/> E <code>bge a0, t0, getret
bge a1, t1, getret</code> | <input type="checkbox"/> H <code>ble a0, t0, getret
ble a1, t1, getret</code> |
| <input type="checkbox"/> C <code>bgt a0, t0, getret
bgt a1, t1, getret</code> | <input type="checkbox"/> F <code>blt a0, t0, getbad
blt a1, t1, getbad</code> | <input type="checkbox"/> I <code>bgt a0, t0, getbad
bgt a1, t1, getbad</code> |

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` ?

- | | | |
|---|---|---|
| <input type="checkbox"/> A <code>mul a1, a1, a0
sll a1, a1, t1</code> | <input type="checkbox"/> D <code>mul a1, a1, t1
add a1, a1, a0</code> | <input type="checkbox"/> G <code>add a1, a1, a0
sll a1, a1, t1</code> |
| <input type="checkbox"/> B <code>add a1, a1, a0
mul a1, a1, t0</code> | <input type="checkbox"/> E <code>sll a1, a1, t0
add a1, a1, a0</code> | <input type="checkbox"/> H <code>add a1, a1, a0
sll a1, a1, t0</code> |
| <input type="checkbox"/> C <code>add a1, a1, a0
mul a1, a1, t1</code> | <input type="checkbox"/> F <code>sll a1, a1, t1
add a1, a1, a0</code> | <input type="checkbox"/> I <code>mul a1, a1, t0
add a1, a1, a0</code> |

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-êre utilisée pour remplacer ??? dans l'instruction « `addi sp, sp, ???` » ?

- | | | | | | | | |
|--------------------------------|-------------------------------|--------------------------------|------------------------------|--------------------------------|---------------------------------|-------------------------------|--------------------------------|
| <input type="checkbox"/> A -32 | <input type="checkbox"/> C -4 | <input type="checkbox"/> E 128 | <input type="checkbox"/> G 8 | <input type="checkbox"/> I -64 | <input type="checkbox"/> K -128 | <input type="checkbox"/> M 32 | <input type="checkbox"/> O -16 |
| <input type="checkbox"/> B 2 | <input type="checkbox"/> D 4 | <input type="checkbox"/> F -8 | <input type="checkbox"/> H 1 | <input type="checkbox"/> J 64 | <input type="checkbox"/> L 16 | <input type="checkbox"/> N -1 | <input type="checkbox"/> P -2 |

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

- | | | | |
|--|--|--|--|
| <input type="checkbox"/> A <code>lw sp, 8(s0)</code> | <input type="checkbox"/> E <code>lw sp, 8(a0)</code> | <input type="checkbox"/> I <code>ld sp, 8(s0)</code> | <input type="checkbox"/> M <code>sw sp, 8(a0)</code> |
| <input type="checkbox"/> B <code>lw s0, 8(sp)</code> | <input type="checkbox"/> F <code>sd s0, 8(sp)</code> | <input type="checkbox"/> J <code>sd a0, 8(sp)</code> | <input type="checkbox"/> N <code>sd sp, 8(a0)</code> |
| <input type="checkbox"/> C <code>lw a0, 8(sp)</code> | <input type="checkbox"/> G <code>sw sp, 8(s0)</code> | <input type="checkbox"/> K <code>sd sp, 8(s0)</code> | <input type="checkbox"/> O <code>ld a0, 8(sp)</code> |
| <input type="checkbox"/> D <code>ld s0, 8(sp)</code> | <input type="checkbox"/> H <code>sw a0, 8(sp)</code> | <input type="checkbox"/> L <code>ld sp, 8(a0)</code> | <input type="checkbox"/> P <code>sw s0, 8(sp)</code> |

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.

- | | | | |
|------------------------------|------------------------------|-------------------------------|-------------------------------|
| <input type="checkbox"/> A 6 | <input type="checkbox"/> D 5 | <input type="checkbox"/> G 64 | <input type="checkbox"/> J 8 |
| <input type="checkbox"/> B 4 | <input type="checkbox"/> E 1 | <input type="checkbox"/> H 0 | <input type="checkbox"/> K 16 |
| <input type="checkbox"/> C 3 | <input type="checkbox"/> F 2 | <input type="checkbox"/> I 7 | <input type="checkbox"/> 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 t1, l(t1)
ld a3, d(t1)

ebreak

```

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

- | | | | | | | | |
|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| <input type="checkbox"/> A 0x00 | <input type="checkbox"/> C 0x10 | <input type="checkbox"/> E 0x20 | <input type="checkbox"/> G 0x30 | <input type="checkbox"/> I 0x40 | <input type="checkbox"/> K 0xD0 | <input type="checkbox"/> M 0xE0 | <input type="checkbox"/> O 0xF0 |
| <input type="checkbox"/> B 0x08 | <input type="checkbox"/> D 0x18 | <input type="checkbox"/> F 0x28 | <input type="checkbox"/> H 0x38 | <input type="checkbox"/> J 0x48 | <input type="checkbox"/> L 0xD8 | <input type="checkbox"/> N 0xE8 | <input type="checkbox"/> P 0xF8 |

Question 15 Même question pour le registre `a2` ?

- | | | | | | | | |
|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| <input type="checkbox"/> A 0x00 | <input type="checkbox"/> C 0x10 | <input type="checkbox"/> E 0x20 | <input type="checkbox"/> G 0x30 | <input type="checkbox"/> I 0x40 | <input type="checkbox"/> K 0xD0 | <input type="checkbox"/> M 0xE0 | <input type="checkbox"/> O 0xF0 |
| <input type="checkbox"/> B 0x08 | <input type="checkbox"/> D 0x18 | <input type="checkbox"/> F 0x28 | <input type="checkbox"/> H 0x38 | <input type="checkbox"/> J 0x48 | <input type="checkbox"/> L 0xD8 | <input type="checkbox"/> N 0xE8 | <input type="checkbox"/> P 0xF8 |

Question 16 Même question pour le registre `a3` ?

- | | | | | | | | |
|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| <input type="checkbox"/> A 0x00 | <input type="checkbox"/> C 0x10 | <input type="checkbox"/> E 0x20 | <input type="checkbox"/> G 0x30 | <input type="checkbox"/> I 0x40 | <input type="checkbox"/> K 0xD0 | <input type="checkbox"/> M 0xE0 | <input type="checkbox"/> O 0xF0 |
| <input type="checkbox"/> B 0x08 | <input type="checkbox"/> D 0x18 | <input type="checkbox"/> F 0x28 | <input type="checkbox"/> H 0x38 | <input type="checkbox"/> J 0x48 | <input type="checkbox"/> L 0xD8 | <input type="checkbox"/> N 0xE8 | <input type="checkbox"/> 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 $8.75_{(10)}$?

- | | | | |
|---|--|--|---|
| <input type="checkbox"/> A $1.00011_{(2)} \times 2^3$ | <input type="checkbox"/> C $1.0011_{(2)} \times 2^{-3}$ | <input type="checkbox"/> E $1.0011_{(2)} \times 2^2$ | <input type="checkbox"/> G $1.00011_{(2)} \times 2^2$ |
| <input type="checkbox"/> B $1.0011_{(2)} \times 2^{-2}$ | <input type="checkbox"/> D $1.00011_{(2)} \times 2^{-3}$ | <input type="checkbox"/> F $1.00011_{(2)} \times 2^{-2}$ | <input type="checkbox"/> H $1.0011_{(2)} \times 2^3$ |

Question 18 Lequel des flottants IEEE 754 simple-précision suivants représente le nombre -10.125054 ?

- | | | | |
|---------------------------------------|---------------------------------------|---------------------------------------|---------------------------------------|
| <input type="checkbox"/> A 0xC2520913 | <input type="checkbox"/> C 0xC1220039 | <input type="checkbox"/> E 0x42520913 | <input type="checkbox"/> G 0xC2220913 |
| <input type="checkbox"/> B 0xC1520123 | <input type="checkbox"/> D 0x4152FFFF | <input type="checkbox"/> F 0x42229912 | <input type="checkbox"/> 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.

- | | | |
|--|---|---|
| <input type="checkbox"/> A 256 | <input type="checkbox"/> E 255 | <input type="checkbox"/> I 31 |
| <input type="checkbox"/> B 127 | <input type="checkbox"/> F 31.9999980926513671875 | <input type="checkbox"/> J 255.9999847412109375 |
| <input type="checkbox"/> C 127.99999237060546875 | <input type="checkbox"/> G 64 | <input type="checkbox"/> K 32 |
| <input type="checkbox"/> D 63.999996185302734375 | <input type="checkbox"/> H 63 | <input type="checkbox"/> 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.

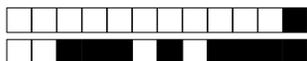
Les entrées du circuit sont: les 64 bits qui codent le nombre et 3 bits qui représentent l'arrondi demandé. Les sorties du circuit sont: les 64 bits qui codent le résultat et 4 bits de drapeaux qui codent une exception des IEEE 754 possible : opération invalide, débordement (*overflow*), débordement 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 ?

- | | | | |
|--|---|--|--|
| <input type="checkbox"/> A 2^{64+4} | <input type="checkbox"/> D $2 \times (64 + 4)$ | <input type="checkbox"/> G $2 \times (64 + 3)$ | <input type="checkbox"/> J $(64 + 3 + 64 + 4)^2$ |
| <input type="checkbox"/> B $(64 + 3)^2$ | <input type="checkbox"/> E $2 \times (64 + 3 + 64 + 4)$ | <input type="checkbox"/> H $(64 + 3)^2$ | <input type="checkbox"/> K $64 + 3$ |
| <input type="checkbox"/> C $2^{64+3+64+4}$ | <input type="checkbox"/> F 2^{64+3} | <input type="checkbox"/> I $64 + 4$ | <input type="checkbox"/> L $64 + 3 + 64 + 4$ |

brouillon

brouillon



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

0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8
9	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 : A B C D E F G H

Question 2 : A B C D E F G H

Question 3 : A B C D E F G H

Question 4 : A B C D E F G H I

Question 5 : A B C D E F G H I

Question 6 : A B C D E F G H I

Question 7 : A B C D E F G H I

Question 8 : A B C D E F G H I J K L

Question 9 : A B C D E F G H I

Question 10 : A B C D E F G H I

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

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

Question 13 : A B C D E F G H I J K L

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

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

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

Question 17 : A B C D E F G H

Question 18 : A B C D E F G H

Question 19 : A B C D E F G H I J K L

Question 20 : A B C D E F 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]	
sraiw	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 ≥	B	0x63	0x5		if(rs1 ≥ rs2) PC += imm	
bltu	Branch < (U)	B	0x63	0x6		if(rs1 < rs2) PC += imm	
bgeu	Branch ≥ (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

	31	25	24	20	19	15	14	12	11	7	6	0
R	fn7			rs2	rs1	fn3	rd			opcode		
I	imm[11:0]				rs1	fn3	rd			opcode		
S	imm[11:5]			rs2	rs1	fn3	imm[4:0]			opcode		
B	imm[12 10:5]			rs2	rs1	fn3	imm[4:1 11]			opcode		
U	imm[31:12]						rd			opcode		
J	imm[20 10:1 11 19:12]						rd			opcode		

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 ≠ 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 ≠ zero
blez rs, lab	bge x0, rs, lab	Branch if ≤ zero
bgez rs, lab	bge rs, x0, lab	Branch if ≥ 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 <
bgtu rs, rt, lab	bltu rt, rs, lab	Branch if > (U)
bleu rs, rt, lab	bgeu rt, rs, lab	Branch if ≤ (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 <i>n</i>	Des octets à 0 pour aligner sur 2^n
.ascii <i>s</i>	Code ascii de chacun des caractères de <i>s</i>
.asciz <i>s</i>	Code ascii de chacun des caractères de <i>s</i> suivi de '\0'
.byte <i>n</i>	Une ou plusieurs valeurs sur un octet
.data	Travaille dans la section Data
.double <i>n</i>	Une ou plusieurs valeurs flottantes double précision
.dword <i>n</i>	Une ou plusieurs valeurs sur 8 octets (double mot)
.eqv <i>s, n</i>	Attribue la valeur <i>n</i> au symbole <i>s</i>
.float <i>n</i>	Une ou plusieurs valeurs flottantes simple précision
.half <i>n</i>	Une ou plusieurs valeurs sur 2 octets (demi mot)
.space <i>n</i>	<i>n</i> octets à 0
.string <i>s</i>	Alias pour .ascii
.text	Travaille dans la section Text
.word <i>n</i>	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 chaîne 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	$\approx -1.70 \times 10^{38}$	$\approx 1.70 \times 10^{38}$