

INF2171 — Organisation des ordinateurs et assembleur

Examen final hiver 2024

Jean Privat & Hugo Leblanc — Dimanche 28 avril

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 Indiquez le bon ordre de grandeur pour les différents niveaux d'un ordinateur.

- A Assembleur < Microarchitecture < Transistors < Circuits Logiques
- B Microarchitecture < Transistors < Assembleur < Circuits Logiques
- C Circuits Logiques < Microarchitecture < Transistors < Assembleur
- D Transistors < Microarchitecture < Circuits Logiques < Assembleur
- E Microarchitecture < Assembleur < Transistors < Circuits Logiques
- F Assembleur < Transistors < Circuits Logiques < Microarchitecture
- G Circuits Logiques < Microarchitecture < Assembleur < Transistors
- H Transistors < Circuits Logiques < Microarchitecture < Assembleur
- I Aucune de ces réponses.

Question 2 En RISC-V, quand l'instruction `j` est exécutée, qu'arrive-t-il avec l'adresse de l'instruction courante ?

- A Elle est empilée dans la pile à l'adresse indiquée par `sp`.
- B Elle est écrite dans le registre `sp`.
- C Elle est empilée sur le tas à l'adresse indiquée par `SBreak`.
- D Elle est écrite dans le registre `a0`.
- E Elle est écrite dans le registre `s0`.
- F Elle est écrite dans le registre `ra`.
- G Aucune de ces réponses.

Question 3 ♣ En RISC-V, quelles sont les caractéristiques des interruptions matérielles ? Indiquez toutes les bonnes réponses.

- A Elles branchent automatiquement sur une gérante d'interruption
- B La gérante est responsable de sauvegarder et restaurer les registres `a0` à `a7` et `t0` à `t6` écrasés
- C Elles sont déclenchées de manière asynchrones au programme
- D La gérante est responsable de restaurer le compteur ordinal à partir du registre `ra`
- E Elles sont numérotées pour catégorisation
- F Elles surviennent aussi quand une exception CPU (faute) est levée par le programme
- G Elles nécessitent une attente active (*polling*)
- H La gérante est responsable de sauvegarder le compteur ordinal original dans le registre `ra`
- I Quand elles surviennent, le clavier se blo

brouillon

2 Programme mystère

Soit le programme mystérieux suivant :

```

    la ra, ici
    li a7, 11 # PrintChar
    li a0, 'A'
    ecall
    li a0, 'B'
    ret
la:
    la ra, ici
    li a0, 'C'
    ecall
    li a0, 'D'
    la ra, bas
    ret
ici:
    la ra, bas
    li a0, 'E'
    ecall
    li a0, 'F'
    call la
    li a0, 'G'
    ret
bas:
    li a0, 'H'
    ecall
    li a0, 'I'
    ret

```

Question 4 Quelle est la deuxième lettre affichée par ce programme mystérieux ?

A
 B
 C

D
 E
 F

G
 H
 I

J *Aucune de ces réponses*

Question 5 Quelle est la troisième lettre affichée par ce programme mystérieux ?

A
 B
 C

D
 E
 F

G
 H
 I

J *Aucune de ces réponses*

Question 6 Quelle est la quatrième lettre affichée par ce programme mystérieux ?

A
 B
 C

D
 E
 F

G
 H
 I

J *Aucune de ces réponses*

Question 7 Quelle est la cinquième lettre affichée par ce programme mystérieux ?

A
 B
 C

D
 E
 F

G
 H
 I

J *Aucune de ces réponses*

3 Routines, matrices, et pile

Soit le programme incomplet suivant qui lit de l'utilisateur un numéro et affiche le nombre de cases remplies de la colonne correspondante pour une matrice en mémoire. Une fois complété, si l'on saisit « 2 », le programme affiche 3 cellules remplies avec la colonne 2. Le programme utilise les routines de la librairie « `libs.s` ». Vous pouvez assumer qu'elle fait partie de l'assemblage.

```

.data
msg:  .string " cellules remplies avec la colonne "
mat:  .ascii "..#.."
      .ascii "#.#.."
      .ascii "...##"
      .ascii "####"
      .eqv rempli, '#'
      .text
      call readInt
      li a1, 4
      li a2, 5
      # TODO 1 *****
      call exit
# prt : Affiche le nombre de cellules remplies d'une colonne
# a0 : Le numéro de la colonne de la matrice à traverser
# a1 : Le hauteur de la matrice
# a2 : La largeur de la matrice
# a3 : L'adresse de la matrice
# Aucun retour
prt:
      # Prologue
      addi sp, sp, ????      # TODO 2 *****
      # [...]

      mv s0, a0
      li s1, 0 # Nombre de cases remplies rencontrées
      mv s2, a1
      mv s3, a2
      mv s4, a3
      # calcul de s4 = Adresse de la première case à vérifier:
      # TODO 3 *****
loop:  beqz s2, prtfin
      li t0, rempli
      # TODO 4 *****      # if (case != '#') {
      addi s1, s1, 1          #   compte++;
                              #}

prtпас: # TODO 5 *****
      j loop

prtfin: mv a0, s1
      call printInt
      la a0, msg
      call printString
      mv a0, s0
      call printInt
      # Épilogue
      # [...]
      ret

```

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

- | | | | |
|---|---|---|---|
| <input type="checkbox"/> A <code>ld a3, mat
j prt</code> | <input type="checkbox"/> D <code>la a3, mat
j prt</code> | <input type="checkbox"/> G <code>lb a3, mat
j prt</code> | <input type="checkbox"/> J <code>ld a3, mat
call prt</code> |
| <input type="checkbox"/> B <code>la a3, mat
call prt</code> | <input type="checkbox"/> E <code>li a3, mat
call prt</code> | <input type="checkbox"/> H <code>li a3, mat
j prt</code> | <input type="checkbox"/> K <code>lb a3, mat
jalr prt</code> |
| <input type="checkbox"/> C <code>la a3, mat
jalr prt</code> | <input type="checkbox"/> F <code>li a3, mat
jalr prt</code> | <input type="checkbox"/> I <code>ld a3, mat
jalr prt</code> | <input type="checkbox"/> L <code>lb a3, mat
call prt</code> |

Question 9 Quelle valeur devrait être utilisée pour remplacer `????` dans l'instruction « `addi sp, sp, ????` » dans le `TODO 2` ?

- | | | | | | |
|--------------------------------|--------------------------------|--------------------------------|-------------------------------|--------------------------------|--|
| <input type="checkbox"/> A -8 | <input type="checkbox"/> D 0 | <input type="checkbox"/> G -48 | <input type="checkbox"/> J 24 | <input type="checkbox"/> M -16 | <input type="checkbox"/> P <i>Aucune
de ces
réponses</i> |
| <input type="checkbox"/> B -40 | <input type="checkbox"/> E -56 | <input type="checkbox"/> H 48 | <input type="checkbox"/> K 56 | <input type="checkbox"/> N 16 | |
| <input type="checkbox"/> C -24 | <input type="checkbox"/> F 8 | <input type="checkbox"/> I -32 | <input type="checkbox"/> L 40 | <input type="checkbox"/> O 32 | |

Question 10 Quelle instruction met `s4` à la première adresse que la routine devra vérifier, à la place de `TODO 3` ?

- | | | |
|---|---|--|
| <input type="checkbox"/> A <code>add s4, s0, mat</code> | <input type="checkbox"/> D <code>addi s4, s4, a0</code> | <input type="checkbox"/> G <code>add s4, s4, s0</code> |
| <input type="checkbox"/> B <code>mul s4, s4, s0</code> | <input type="checkbox"/> E <code>mul s4, s2, s3</code> | <input type="checkbox"/> H <code>slli s4, s4, 2</code> |
| <input type="checkbox"/> C <code>la s4, mat</code> | <input type="checkbox"/> F <code>addi s4, s4, 4</code> | <input type="checkbox"/> I <code>ld s4, mat(0)</code> |

Question 11 Que doit-on mettre pour compléter la vérification du caractère dans la routine `prt`, à la place de `TODO 4` ?

- | | | |
|--|---|--|
| <input type="checkbox"/> A <code>lb t1, 0(s4)
bne t0, t1, loop</code> | <input type="checkbox"/> D <code>ld t1, 0(s4)
beq t0, t1, prtfin</code> | <input type="checkbox"/> G <code>ld t1, 0(s4)
beq t0, t1, prt pas</code> |
| <input type="checkbox"/> B <code>ld t1, 0(s4)
bne t0, t1, prt pas</code> | <input type="checkbox"/> E <code>ld t1, 0(s4)
j loop</code> | <input type="checkbox"/> H <code>lb t1, 0(s4)
j prt pas</code> |
| <input type="checkbox"/> C <code>lb t1, 0(s4)
bne t0, t1, prt pas</code> | <input type="checkbox"/> F <code>ld t1, 0(s4)
beq t0, t1, loop</code> | <input type="checkbox"/> I <code>ld t1, 0(s4)
beq t0, t1, prtfin</code> |

Question 12 Que doit-on mettre pour préparer à la prochaine boucle dans la routine `prt`, à la place de `TODO 5` ?

- | | | |
|--|--|---|
| <input type="checkbox"/> A <code>addi s2, s2, -1
add s4, s4, s3</code> | <input type="checkbox"/> D <code>add s2, s2, s3
add s4, s4, s2</code> | <input type="checkbox"/> G <code>addi s2, s2, 1
add s4, s4, s2</code> |
| <input type="checkbox"/> B <code>add s2, s3, s0
add s4, s4, s2</code> | <input type="checkbox"/> E <code>add s2, s2, s0
add s4, s4, s3</code> | <input type="checkbox"/> H <code>addi s2, s2, 1
add s4, s4, s3</code> |
| <input type="checkbox"/> C <code>add s2, s2, s0
add s4, s4, s2</code> | <input type="checkbox"/> F <code>addi s2, s2, -1
add s4, s4, s2</code> | <input type="checkbox"/> I <code>add s2, s2, s3
add s4, s4, s3</code> |

4 Allocation et récursivité

Soit le programme suivant :

```

.data
# Appel système
.eqv Sbrk, 9
# Structure maillon
.eqv mChar, 0 # byte
.eqv mLeft, 8 # dword
.eqv mRight, 16 # dword
.eqv mLen, 24 # taille

str: .ascii "ABCDEFGHJKLMNQP"

.text
la a0, str
li a1, 2
jal deploy
ld a0, 8(a0)
ld s0, 0(a0)

la a0, str
li a1, 4
jal deploy
ld a0, 8(a0)
ld a0, 16(a0)
ld s1, 0(a0)

la a0, str
li a1, 15
jal deploy
ld a0, 8(a0)
ld a0, 16(a0)
ld a0, 16(a0)
ld a0, 8(a0)
ld a0, 16(a0)
ld a0, 8(a0)
ld a0, 16(a0)
ld a0, 16(a0)
ld s2, 0(a0)

ebreak
# ...

# a0: chaîne à deployer
# a1: nombre de caractères de la chaîne a1
# retour: a0 adresse du maillon racine
deploy:
    addi sp, sp, -32
    sd ra, 0(sp)
    sd s0, 8(sp)
    sd s1, 16(sp)
    sd s2, 24(sp)

    mv s0, a0
    mv s1, a1

    li a0, 0
    blez s1, deploy_fin

    li a0, mLen
    li a7, Sbrk
    ecall
    mv s2, a0
    lbu a0, 0(s0)
    sd a0, mChar(s2)

    addi a0, s0, 1
    addi a1, s1, -1
    jal deploy
    sd a0, mLeft(s2)

    addi a0, s0, 2
    addi a1, s1, -2
    jal deploy
    sd a0, mRight(s2)

    mv a0, s2
deploy_fin:
    ld ra, 0(sp)
    ld s0, 8(sp)
    ld s1, 16(sp)
    ld s2, 24(sp)
    addi sp, sp, 32
    ret

```

Question 13 Quelle est la valeur de l'octet de poids faible de `s0` lors du `ebreak` ?

- | | | | |
|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| <input type="checkbox"/> A 0x41 | <input type="checkbox"/> E 0x45 | <input type="checkbox"/> I 0x49 | <input type="checkbox"/> M 0x4D |
| <input type="checkbox"/> B 0x42 | <input type="checkbox"/> F 0x46 | <input type="checkbox"/> J 0x4A | <input type="checkbox"/> N 0x4E |
| <input type="checkbox"/> C 0x43 | <input type="checkbox"/> G 0x47 | <input type="checkbox"/> K 0x4B | <input type="checkbox"/> O 0x4F |
| <input type="checkbox"/> D 0x44 | <input type="checkbox"/> H 0x48 | <input type="checkbox"/> L 0x4C | <input type="checkbox"/> P 0x50 |

Question 14 Quelle est la valeur de l'octet de poids faible de `s1` lors du `ebreak` ?

- | | | | |
|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| <input type="checkbox"/> A 0x41 | <input type="checkbox"/> E 0x45 | <input type="checkbox"/> I 0x49 | <input type="checkbox"/> M 0x4D |
| <input type="checkbox"/> B 0x42 | <input type="checkbox"/> F 0x46 | <input type="checkbox"/> J 0x4A | <input type="checkbox"/> N 0x4E |
| <input type="checkbox"/> C 0x43 | <input type="checkbox"/> G 0x47 | <input type="checkbox"/> K 0x4B | <input type="checkbox"/> O 0x4F |
| <input type="checkbox"/> D 0x44 | <input type="checkbox"/> H 0x48 | <input type="checkbox"/> L 0x4C | <input type="checkbox"/> P 0x50 |

Question 15 Quelle est la valeur de l'octet de poids faible de `s2` lors du `ebreak` ?

- | | | | |
|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| <input type="checkbox"/> A 0x41 | <input type="checkbox"/> E 0x45 | <input type="checkbox"/> I 0x49 | <input type="checkbox"/> M 0x4D |
| <input type="checkbox"/> B 0x42 | <input type="checkbox"/> F 0x46 | <input type="checkbox"/> J 0x4A | <input type="checkbox"/> N 0x4E |
| <input type="checkbox"/> C 0x43 | <input type="checkbox"/> G 0x47 | <input type="checkbox"/> K 0x4B | <input type="checkbox"/> O 0x4F |
| <input type="checkbox"/> D 0x44 | <input type="checkbox"/> H 0x48 | <input type="checkbox"/> L 0x4C | <input type="checkbox"/> P 0x50 |

Question 16 Lors du 3e appel à `deploy` du programme principal (l'appel avec `a1=15`), quelle est la quantité minimale de pile nécessaire (afin de se rendre et pouvoir exécuter les appels les plus profonds).

- | | | | |
|---------------------------------------|--|---|--|
| <input type="checkbox"/> A 16 octets | <input type="checkbox"/> D 512 octets | <input type="checkbox"/> G 4096 octets | <input type="checkbox"/> J 65546 octets |
| <input type="checkbox"/> B 32 octets | <input type="checkbox"/> E 1024 octets | <input type="checkbox"/> H 16384 octets | <input type="checkbox"/> K 102176 octets |
| <input type="checkbox"/> C 256 octets | <input type="checkbox"/> F 3193 octets | <input type="checkbox"/> I 32768 octets | <input type="checkbox"/> L 131072 octets |

brouillon

5 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 $12.25_{(10)}$?

- | | | | |
|--|---|--|--|
| <input type="checkbox"/> A $1.00001_{(2)} \times 2^2$ | <input type="checkbox"/> C $1.110001_{(2)} \times 2^{-3}$ | <input type="checkbox"/> E $1.10001_{(2)} \times 2^3$ | <input type="checkbox"/> G $1.110001_{(2)} \times 2^2$ |
| <input type="checkbox"/> B $1.00001_{(2)} \times 2^{-3}$ | <input type="checkbox"/> D $1.110001_{(2)} \times 2^{-2}$ | <input type="checkbox"/> F $1.110001_{(2)} \times 2^3$ | <input type="checkbox"/> H $1.00001_{(2)} \times 2^{-2}$ |

Question 18 Lequel des flottants IEEE 754 simple-précision suivants représente le nombre $28.125_{(10)}$?

- | | | | |
|---------------------------------------|---------------------------------------|---------------------------------------|---------------------------------------|
| <input type="checkbox"/> A 0x41E10000 | <input type="checkbox"/> C 0xC1E20000 | <input type="checkbox"/> E 0x41E00000 | <input type="checkbox"/> G 0xC1610000 |
| <input type="checkbox"/> B 0x41610000 | <input type="checkbox"/> D 0x41E20000 | <input type="checkbox"/> F 0xC1E10000 | <input type="checkbox"/> H 0xC1E30000 |

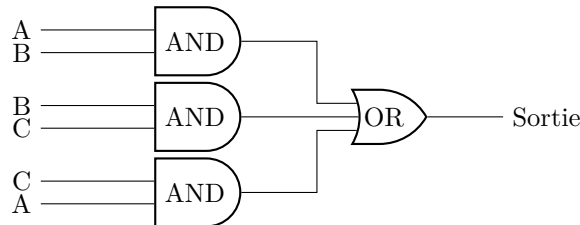
Question 19 ♣ Soit un nombre flottant IEEE 754 simple-précision codé en hexadécimal par $0xC0??????$ (l'octet de poids fort est $0xC0$, les autres octets sont inconnus). Parmi les propositions suivantes, lesquelles sont des valeurs possibles de ce nombre ?

Note : tous les nombres proposés sont exactement représentables en IEEE 754 simple-précision.

- | | | |
|----------------------------------|---------------------------------------|--------------------------------|
| <input type="checkbox"/> A -8.25 | <input type="checkbox"/> E 125 | <input type="checkbox"/> I 0 |
| <input type="checkbox"/> B -125 | <input type="checkbox"/> F 7.5 | <input type="checkbox"/> J -8 |
| <input type="checkbox"/> C -3.5 | <input type="checkbox"/> G -7.5234375 | <input type="checkbox"/> K -6 |
| <input type="checkbox"/> D 3.5 | <input type="checkbox"/> H -3 | <input type="checkbox"/> L -16 |

6 Circuits logiques

Question 20 En étudiant le circuit logique suivant et en établissant sa table de vérité, combien de combinaisons des entrées A, B et C résultent en une sortie vraie ?



- | | | | | |
|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|
| <input type="checkbox"/> A 1 | <input type="checkbox"/> C 3 | <input type="checkbox"/> E 5 | <input type="checkbox"/> G 7 | <input type="checkbox"/> I 9 |
| <input type="checkbox"/> B 2 | <input type="checkbox"/> D 4 | <input type="checkbox"/> F 6 | <input type="checkbox"/> H 8 | <input type="checkbox"/> J 0 |

brouillon

brouillon



Feuille de réponses, final hiver 2024 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	I
---	---	---	---	---	---	---	---	---
- Question 2 :

A	B	C	D	E	F	G
---	---	---	---	---	---	---
- Question 3 :

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

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

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

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

A	B	C	D	E	F	G	H	I	J
---	---	---	---	---	---	---	---	---	---
- 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	J	K	L	M	N	O	P
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
- Question 10 :

A	B	C	D	E	F	G	H	I
---	---	---	---	---	---	---	---	---
- Question 11 :

A	B	C	D	E	F	G	H	I
---	---	---	---	---	---	---	---	---
- Question 12 :

A	B	C	D	E	F	G	H	I
---	---	---	---	---	---	---	---	---
- Question 13 :

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

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

