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 bo | n ordre de grandeur | pour les différents | niveaux d'un | ordinateur |
|------------|----------------|---------------------|---------------------|--------------|------------|
|------------|----------------|---------------------|---------------------|--------------|------------|

- A Assembleur < Microarchitecture < Transistors < Circuits Logiques
- B Microarchitecture < Transistors < Assembleur < Circuits Logiques
- Circuits Logiques < Microarchitecture < Transistors < Assembleur
- D Transistors < Microarchitecture < Circuits Logiques < Assembleur
- 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 indiqué par sp.
- B Elle est écrite dans le registre sp.
- C Elle est empilée sur le tas à l'adresse indiqué par SBreak.
- D Elle est écrite dans le registre a0.
- 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

2 Programme mystère

Soit le programme mystérieux suivant : la ra, ici li a7, 11 # PrintCharli 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 'F' li a0, call la li a0, 'G' bas: li a0, ecall li a0, 'I' ret Question 4 Quelle est la deuxième lettre affichée par ce programme mystérieux? Aucune de ces \mathbf{E} ВВ réponses Question 5 Quelle est la troisième lettre affichée par ce programme mystérieux? A A B B C C Aucune de ces r'eponsesQuestion 6 Quelle est la quatrième lettre affichée par ce programme mystérieux? A AB BC C J | Aucune de ces r'eponsesQuelle est la cinquième lettre affichée par ce programme mystérieux? Question 7 Aucune de ces В \mathbf{E} r'eponses

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.

```
.string " cellules remplies avec la colonne "
msg:
        .ascii "..#.."
mat:
        .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++;
                                        #}
prtpas: # 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?

 \boxed{A} 1d a3, mat j prt

la a3, mat
j prt

Ib a3, mat j prt

J ld a3, mat
call prt

B la a3, mat call prt

E li a3, mat call prt

 $_{\scriptsize \begin{tabular}{ll} \begin{tabular}{ll}$

lb a3, mat
jalr prt

 \square la a3, mat jalr prt

F li a3, mat jalr prt

I ld a3, mat jalr prt

 \square lb a3, mat call prt

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

G -48 H 48 I -32

M -16

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

A add s4, s0, mat

B mul s4, s4, s0 C la s4, mat

D addi s4, s4, a0 E mul s4, s2, s3 F addi s4, s4, 4

G add s4, s4, s0 H slli s4, s4, 2 I ld s4, mat(0)

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

lb t1, 0(s4)
bne t0, t1, loop

D ld t1, 0(s4) beq t0, t1, prtfin

G ld t1, 0(s4) beq t0, t1, prtpas

B ld t1, 0(s4)
bne t0, t1, prtpas

E ld t1, 0(s4) j loop

Ib t1, 0(s4)
j prtpas

D lb t1, 0(s4)
bne t0, t1, prtpas

F ld t1, 0(s4) beq t0, t1, loop

ld t1, 0(s4)
beq t0, t1, prtfin

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

A addi s2, s2, -1 add s4, s4, s3

D add s2, s2, s3 add s4, s4, s2

G addi s2, s2, 1 add s4, s4, s2

B add s2, s3, s0 add s4, s4, s2

E add s2, s2, s0 add s4, s4, s3

H addi s2, s2, 1 add s4, s4, s3

C add s2, s2, s0 add s4, s4, s2

 \mathbb{F} addi s2, s2, -1 add s4, s4, s2

add s2, s2, s3 add s4, s4, s3

4 Allocation et récursivité

Soit le programme suivant :

```
.data
                                                # a0: chaine à deployer
        # Appel système
                                                 # a1: nombre de caractères de la chaîne a1
        .eqv Sbrk, 9
                                                 # retour: a0 adresse du maillon racine
        # Structure maillon
                                                deploy:
        .eqv mChar, 0 # byte
                                                         addi sp, sp, -32
        .eqv mLeft, 8 # dword
                                                         sd ra, O(sp)
        .eqv mRight, 16 # dword
                                                         sd s0, 8(sp)
                                                         sd s1, 16(sp)
        .eqv mLen, 24 # taille
                                                         sd s2, 24(sp)
       .ascii "ABCDEFGHIJKLMNOP"
str:
                                                         mv s0, a0
        .text
                                                         mv s1, a1
        la a0, str
        li a1, 2
                                                         li a0, 0
        jal deploy
                                                         blez s1, deploy_fin
        ld a0, 8(a0)
        ld s0, 0(a0)
                                                         li a0, mLen
                                                         li a7, Sbrk
        la a0, str
                                                         ecall
                                                         mv s2, a0
        li a1, 4
        jal deploy
                                                         1bu a0, 0(s0)
        ld a0, 8(a0)
                                                         sd a0, mChar(s2)
        ld a0, 16(a0)
        ld s1, 0(a0)
                                                         addi a0, s0, 1
                                                         addi a1, s1, -1
        la a0, str
                                                         jal deploy
                                                         sd a0, mLeft(s2)
        li a1, 15
        jal deploy
        ld a0, 8(a0)
                                                         addi a0, s0, 2
        ld a0, 16(a0)
                                                         addi a1, s1, -2
        ld a0, 16(a0)
                                                         jal deploy
        ld a0, 8(a0)
                                                         sd a0, mRight(s2)
        ld a0, 16(a0)
       ld a0, 8(a0)
                                                         mv a0, s2
        ld a0, 16(a0)
                                                deploy_fin:
        ld a0, 16(a0)
                                                         ld ra, 0(sp)
        ld s2, 0(a0)
                                                         ld s0, 8(sp)
                                                         ld s1, 16(sp)
                                                         ld s2, 24(sp)
                                                         addi sp, sp, 32
        ebreak
        # ...
                                                         ret
```

 $65546 \ {\rm octets}$

102176 octets 131072 octets

| Question 13 | Quelle est la valeur de l'octet de poids lai | ible de su fors du ebr | eak: |
|--|--|--------------------------------------|--------------------------------------|
| A 0x41 B 0x42 C 0x43 D 0x44 | E 0x45 F 0x46 G 0x47 H 0x48 | I 0x49 J 0x4A K 0x4B L 0x4C | M 0x4D N 0x4E O 0x4F P 0x50 |
| Question 14 | Quelle est la valeur de l'octet de poids fai | ble de s1 lors du ebr | eak? |
| A 0x41 B 0x42 C 0x43 D 0x44 Question 15 | E 0x45 F 0x46 G 0x47 H 0x48 Quelle est la valeur de l'octet de poids fai | I 0x49 J 0x4A K 0x4B L 0x4C | M 0x4D N 0x4E 0 0x4F P 0x50 |
| A 0x41 B 0x42 C 0x43 D 0x44 | E 0x45 F 0x46 G 0x47 H 0x48 | I 0x49 J 0x4A K 0x4B L 0x4C | M 0x4D N 0x4E 0 0x4F P 0x50 |
| Question 16 minimale de pile | Lors du 3e appel à deploy du programm e nécessaire (afin de se rendre et pouvoir ex | \ | / · - |

 $4096 \ {\rm octets}$

H 16384 octets I 32768 octets

512 octets

1024 octets

3193 octets

16 octets

32 octets

C 256 octets

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₎?

 $\begin{array}{c|c} \hline A & 1.00001_{(2)} \times 2^2 \\ \hline B & 1.00001_{(2)} \times 2^{-3} \\ \end{array}$

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

0x41E10000 0x41610000

0xC1E20000 0x41E20000

E 0x41E00000 F 0xC1E10000

Question 19 4 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.

-8.25

-125

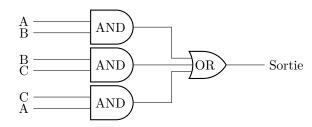
-3.5

3.5

G -7.5234375

Circuits logiques

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?





Feuille de réponses, final hiver 2024 INF2171 — Organisation des ordinateurs et assembleur

| 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 |
| 4 4 4 4 4 4 4 |
| 5 5 5 5 5 5 5 |
| 6 6 6 6 6 6 6 |
| 7 7 7 7 7 7 7 7 |
| 8 8 8 8 8 8 8 |
| 9 9 9 9 9 9 9 |

 \longleftarrow 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 : \overline{A} \overline{B} \overline{C} \overline{D} \overline{E} \overline{F} \overline{G} \overline{H} \overline{I} \overline{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] | |
| srai | Shift Right Arith Imm | Ī | 0x13 | 0x5 | imm[11:6]=0x10 | rd = rs1 >> imm[5:0] | |
| slti | Set Less Than Imm | Ī | 0x13 | 0x2 | | rd = (rs1 < imm)?1:0 | |
| sltiu | Set Less Than Imm (U) | I | 0x13 | 0x3 | | rd = (rs1 < imm)?1:0 | |
| 1b | Load Byte | I | 0x03 | 0x0 | | rd = M[rs1+imm][7:0] | |
| lh | Load Half | Ī | 0x03 | 0x1 | | rd = M[rs1+imm][15:0] | |
| lw | Load Word | Ī | 0x03 | 0x2 | | rd = M[rs1+imm][31:0] | |
| ld | Load Double Word | Ī | 0x03 | 0x3 | | rd = M[rs1+imm][63:0] | RV64I |
| lbu | Load Byte (U) | Ī | 0x03 | 0x4 | | rd = M[rs1+imm][7:0] | 111 0 11 |
| 1bu 1hu | Load Half (U) | Ī | 0x03 | 0x5 | | rd = M[rs1+imm][15:0] | |
| lwu | Load Word (U) | Ī | 0x03 | 0x6 | | rd = M[rs1+imm][63:0] | RV64I |
| sb | Store Byte | S | 0x23 | 0x0 | | M[rs1+imm][7:0] = rs2[7:0] | 10.012 |
| 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 == | В | 0x63 | 0x0 | | if(rs1 == rs2) PC += imm | 100 11 |
| bne | Branch != | В | 0x63 | 0x1 | | if(rs1 != rs2) PC += imm | |
| blt | Branch < | В | 0x63 | 0x1 | | if(rs1 < rs2) PC += imm | |
| bge | Branch ≥ | В | 0x63 | 0x5 | | if(rs1 >= rs2) PC += imm | |
| bltu | Branch < (U) | В | 0x63 | 0x5 | | if(rs1 < rs2) PC += imm | |
| | Branch \geq (U) | В | 0x63 | 0x7 | | if(rs1 >= rs2) PC += imm | |
| bgeu jal | Jump And Link | J | 0x6F | 0 . 7 | | rd = PC+4; PC += imm | |
| jal jalr | Jump And Link Register | J | 0x67 | 0x0 | | rd = PC+4; PC = rs1 + imm | |
| lui | Load Upper Imm | U | 0x67 0x37 | שאש | | rd = imm << 12 | |
| | | | | | | rd = PC + (imm << 12) | |
| auipc | Add Upper Imm to PC | U | 0x17 | ΔΔ | ; mm=0.40 | | |
| ecall | Environment Call | I | 0x73 | 0x0 | imm=0x0 | Appel système | |
| ebreak | Environment Break | I | 0x73 | 0x0 | imm=0x1 | Appel au débogueur | DV20M |
| 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 |
| Ι | imm[11:0] | | rs1 | fn3 | rd | opcode |
| S | imm[11:5] | rs2 | rs1 | fn3 | imm[4:0] | opcode |
| В | imm[12 10:5] | rs2 | rs1 | fn3 | imm[4:1 11] | opcode |
| U | • | 31:12 | - | | rd | opcode |
| J | imm[20 10 | $0:1 11 ^{2}$ | 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 | <pre>auipc rt, H s[bhwd] rd, L(rt)</pre> | 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 \boldsymbol{s} | Code ascii de chacun des caractères de s suivi de '\0' |
| .byte $\it n$ | Une ou plusieurs valeurs sur un octet |
| .data | Travaille dans la section Data |
| .double \boldsymbol{n} | Une ou plusieurs valeurs flotantes double précision |
| .dword \boldsymbol{n} | Une ou plusieurs valeurs sur 8 octets (double mot) |
| .eqv s , n | Attribue la valeur n au symbole s |
| .float \boldsymbol{n} | Une ou plusieurs valeurs flotantes simple précision |
| .half \boldsymbol{n} | Une ou plusieurs valeurs sur 2 octets (demi mot) |
| .space n | n octets à 0 |
| .string \boldsymbol{s} | Alias pour .asciz |
| .text | Travaille dans la section Text |
| .word \boldsymbol{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) |

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 | 65535 |
| | | | oui | -32768 | 32767 |
| 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 | $\approx -1.70 \times 10^{38}$ | $\approx 1.70 \times 10^{38}$ |

Table ASCII

| Dec Hex | | | Dec Hex | | | Dec Hex | | | Dec Hex | | | Dec Hex | | | Dec Hex | | | Dec Hex | | Dec Hex | |
|---------|----|----------|---------|----|-----|---------|----|--------|---------|----|---|---------|----|----------|---------|----|---|---------|---|---------|-----|
| 0 | 00 | NUL '\0' | 16 | 10 | DLE | 32 | 20 | SP , , | 48 | 30 | 0 | 64 | 40 | <u>@</u> | 80 | 50 | Р | 96 60 | 4 | 112 70 | р |
| 1 | 01 | SOH | 17 | 11 | DC1 | 33 | 21 | ! | 49 | 31 | 1 | 65 | 41 | Α | 81 | 51 | Q | 97 61 | а | 113 71 | q |
| 2 | 02 | STX | 18 | 12 | DC2 | 34 | 22 | " | 50 | 32 | 2 | 66 | 42 | В | 82 | 52 | R | 98 62 | b | 114 72 | r |
| 3 | 03 | ETX | 19 | 13 | DC3 | 35 | 23 | # | 51 | 33 | 3 | 67 | 43 | С | 83 | 53 | S | 99 63 | С | 115 73 | s |
| 4 | 04 | EOT | 20 | 14 | DC4 | 36 | 24 | \$ | 52 | 34 | 4 | 68 | 44 | D | 84 | 54 | Т | 100 64 | d | 116 74 | t |
| 5 | 05 | ENQ | 21 | 15 | NAK | 37 | 25 | % | 53 | 35 | 5 | 69 | 45 | Е | 85 | 55 | U | 101 65 | е | 117 75 | u |
| 6 | 06 | ACK | 22 | 16 | SYN | 38 | 26 | & | 54 | 36 | 6 | 70 | 46 | F | 86 | 56 | ٧ | 102 66 | f | 118 76 | V |
| 7 | 07 | BEL '\a' | 23 | 17 | ETB | 39 | 27 | , | 55 | 37 | 7 | 71 | 47 | G | 87 | 57 | W | 103 67 | g | 119 77 | W |
| 8 | 08 | BS '\b' | 24 | 18 | CAN | 40 | 28 | (| 56 | 38 | 8 | 72 | 48 | Н | 88 | 58 | Χ | 104 68 | h | 120 78 | Х |
| 9 | 09 | HT '\t' | 25 | 19 | EM | 41 | 29 |) | 57 | 39 | 9 | 73 | 49 | Ι | 89 | 59 | Υ | 105 69 | i | 121 79 | у |
| 10 | 0A | LF '\n' | 26 | 1A | SUB | 42 | 2A | * | 58 | 3A | : | 74 | 4A | J | 90 | 5A | Z | 106 6A | j | 122 7A | Z |
| 11 | 0B | VT '\v' | 27 | 1B | ESC | 43 | 2B | + | 59 | 3B | ; | 75 | 4B | K | 91 | 5B | Γ | 107 6B | k | 123 7B | { |
| 12 | 0C | FF'\f' | 28 | 1C | FS | 44 | 2C | , | 60 | 3C | < | 76 | 4C | L | 92 | 5C | \ | 108 6C | 1 | 124 7C | 1 |
| 13 | 0D | CR '\r' | 29 | 1D | GS | 45 | 2D | - | 61 | 3D | = | 77 | 4D | М | 93 | 5D |] | 109 6D | m | 125 7D | } |
| 14 | 0E | SO | 30 | 1E | RS | 46 | 2E | | 62 | 3E | > | 78 | 4E | N | 94 | 5E | ^ | 110 6E | n | 126 7E | ~ |
| 15 | 0F | SI | 31 | 1F | US | 47 | 2F | / | 63 | 3F | ? | 79 | 4F | 0 | 95 | 5F | _ | 111 6F | 0 | 127 7F | DEL |