Decoding RISCV instructions

Louis Geoffroy

January 2022

1 Introduction

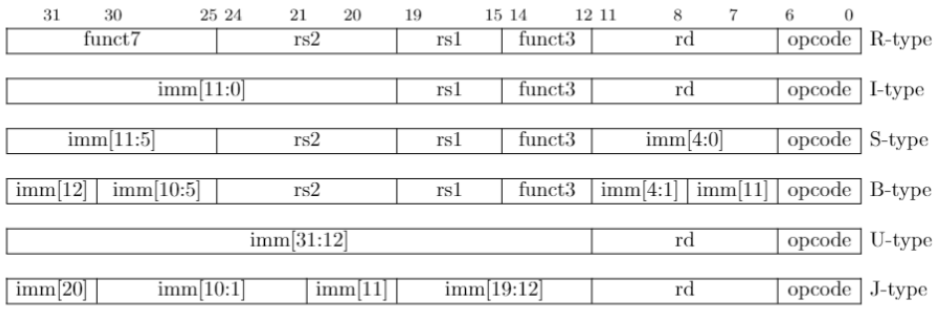


Figure 1: RISC-V inscrutions de base sans extension

1.0.1 R-type :

Les instructions de type R sont des instructions arithmétiques qui utilisent uniquement des registres :

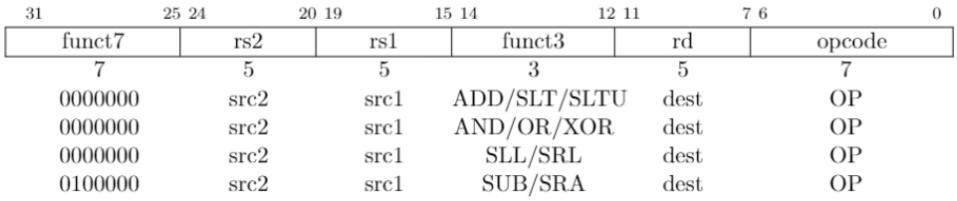


Figure 2: RISC-V inscrutions de type R

- funct7 : donne des informations sur l'instruction effectuée
- rs2 : indique le registre source numéro 2
- rs1 : indique le registre source numéro 1
- funct3 : indique le typé d'opération que l'on doit effectuer
- rd : registrte destination
- opcode = 0110011 : indique que l'on est dans des opérations de type I

Détail des opérations :

- ADD : rd = rs1 + rs2, overflow ignoré
- SUB : rd = rs2-rs1, overflow ignoré
- SLT : rd = 1 si rs1 ¡ rs2, 0 sinon; la comparaison est signée
- SLTU : comme SLT mais la comparaison n'est pas signée

Warning : SLTU rd, r0, rs2 effectue l'opération suivante : rd = 1 si rs2 != 0 sinon 0

- AND, OR, XOR : opération logique associée
- SLL : shift logic left sur rs1 par la valeur contenue dans les 5 bits de poids faible de rs2
- SRL : shift logic right sur rs1 par la valeur contenue dans les 5 bits de poids faible de rs2
- SRA : shift arithmetic right sur rs1 par la valeur contenue dans les 5 bits de poids faible de rs2

Instruction	funct7	funct3	opcode
ADD	0000000	000	0110011
SLT	0000000	010	0110011
SLTU	0000000	011	0110011
AND	0000000	111	0110011
OR	0000000	110	0110011
XOR	0000000	100	0110011
SLL	0000000	001	0110011
SRL	0000000	101	0110011
SUB	0100000	000	0110011
SRA	0100000	101	0110011

1.0.2 I-type :

Ce sont des instrutions où l'opérande 2 est de type immédiat, on considère dans ce cas un immédiat de 12 bits.
Dans tous les cas l'immédiat est sign-extended sur 32 bits.
Les instructions I permettent également d'encoder les instructions de type load qui seront détailler dans la section 1.0.6.

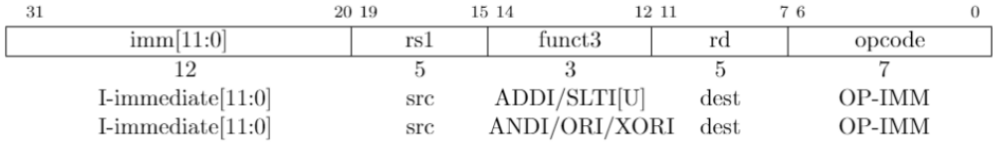


Figure 3: RISC-V inscrutions de type I

- imm[11:0] : immédiat sur 12 bits que l'on va charger dans l'opérande 2
- rs1 : registre source
- funct3 : code le type d'opération effectuée
- rd : registre destination
- opcode : indique que l'on est dans des opérations de type I

Détail des opérations :

- ADDI : ajoute l'immédiat de 12 bits à rs1, overflow ignoré
- SLTI (set less than immediate) : rd = 1 si (rs1 $< \&imm[11:0]$) sinon 0

- SLTIU : idem que SLTI mais dans ce cas comparaison est unsigned, l'immediat est d'abord sign-extend mais est traité comme un entier unsigned
- ANDI, ORI, XORI : opération logique and, or et xor entre rs1 et l'immédiat

Instruction	funct3	opcode
ADDI	000	0010011
SLTI	010	0010011
SLTIU	011	0010011
ANDI	111	0010011
ORI	110	0010011
XORI	100	0010011

Les instructions de type I permettent aussi d'encoder les shifts avec un immédiat. Dans ce cas on va shifter la valeur contenu dans le registre rs1 et la valeur de shift est contenu dans les 5 bits de poids faible de l'immédiat.

31 25	24 20	19 15	14 12	11 7	6 0
imm[11:5]	imm[4:0]	rs1	funct3	rd	opcode
7	5	5	3	5	7
0000000	shamt[4:0]	src	SLLI	dest	OP-IMM
0000000	shamt[4:0]	src	SRLI	dest	OP-IMM
0100000	shamt[4:0]	src	SRAI	dest	OP-IMM

Figure 4: RISC-V inscrutions de type S

- imm[11:5] : Aide à définir l'instruction de shift
- imm[4:0] : valeur du shift
- rs1 : opérande sur laquelle on va appliquer le shift
- funct3 : type de shift
- rd : registre destination
- opcode : indique que l'on est dans des opérations de type I

Détail des instructions :

- SLLI : Shift logique gauche

- SRLI : Shift logique droite
- SRAI :Shift arithmétique droite

Instruction	funct3	opcode
SLLI	001	0010011
SRLI	101	0010011
SRAI	101	0010011

On trouve en plus dans les instructions de type I l'opération NOP qui est définie comme suit :

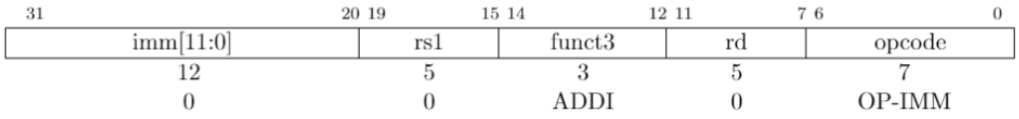


Figure 5: RISC-V NOP inscrution de type I

NOP est encodé comme ADDI r0,r0,0

1.0.3 B-type :

Les instructions de type B sont les instructions de branchement conditionnel. Attention en RISC-V il n'y a pas de delay slot.

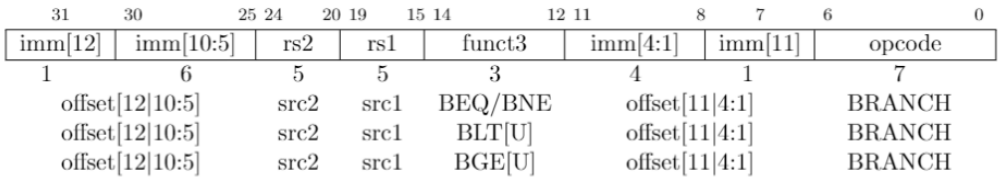


Figure 6: RISC-V inscrutions de type B

Dans chacune des instructions de branchement conditionnelles une comparaison entre le registres rs1 et rs2 est effectuée. De même que précédemment l'offset est sign-extended.

Détail des instructions :

- BEQ : PC = PC + OFFSET si rs1 = rs2
- BNE : PC = PC + OFFSET si rs1 != rs2
- BLT : PC = PC+ OFFSET si rs1 < rs2
- BLTU : idem que BLT mais BLTU utilise une comparaison non signée.
- BGE : PC = PC+ OFFSET si rs1 >= rs2
- BGEU : idem avec comparaison non signée

BGT/BGTU, BLE/BLEU peuvent être obtenu en inversant les opérandes de BLT/BLTU et BGE/BGEU.

Instruction	funct3	opcode
BEQ	000	1100011
BNE	001	1100011
BLT	100	1100011
BGE	101	1100011
BLTU	110	1100011
BGEU	111	1100011

1.0.4 U-type :

Les instructions de types U sont les suivantes, elles permettent de charger un valeur dans les bits de poids fort d'un registre.

Figure 7: RISC-V inscrutions de type U

- imm[31:12] : immédiat sur 20 bits
- rd : registre destination

- opcode : indique que l'on est dans des opérations de type I

Détail des instructions :

- LUI (load upper immediate) : charge les 20 bits de l'immédiat dans les 20 bits de poids fort de rd et complète les bits de poids faible avec des 0.
- AUIPC (add upper immediate to PC) : rd = PC + imm[31:12] 0x000, permet de générer une adresse relative

Instruction	opcode
LUI	0110111
AUIPC	0010111

1.0.5 J-type :

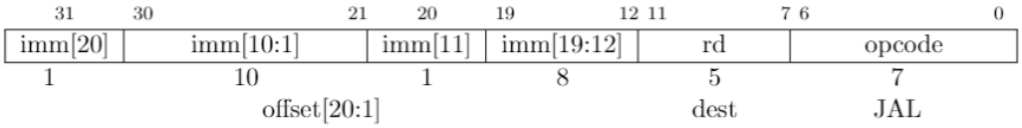


Figure 8: RISC-V inscrutions de type J

- imm[20] imm[10:1] imm[11] imm[19:12] : immédiat signé codant l'offset
- rd : registre desination
- opcode : indique que l'on est dans des opérations de type J

Détail des instructions :
- JAL (jump and link) : l'immédiat encode un offset signé décalé de 2 octets. L'offset est ensuite ajouté à l'adresse contenue dans PC. L'instruction suivant le jump (PC+4) est sauvegardé dans le registre rd.

Instruction	opcode
JAL	1101111

On trouve également l'instruction JALR qui est encodée comme une instruction de type I :

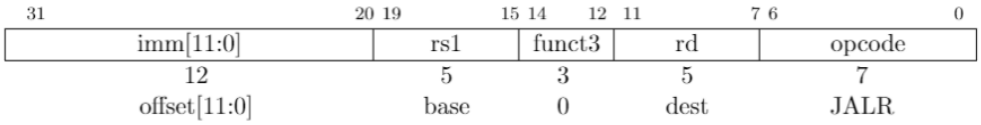


Figure 9: RISC-V inscrution jalr de type I

Dans le cas de l'instruction JALR, l'immédiat est signé.
- rd = PC +4, adresse suivant le branchement.
- PC = [rs1 + immédiat signé étendu sur 32 bits]. Le dernier bit du résultat est mis à 0 pour des raisons d'alignement mémoire.

Instruction	funct3	opcode
JALR	000	1100111

The standard software calling convention uses x1 as the return address register and x5 as an alternate link register.

1.0.6 Acces mémoire :

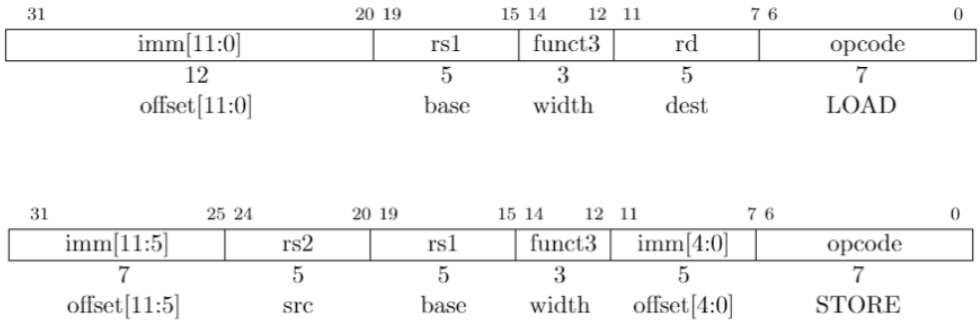


Figure 10: RISC-V inscrutions de type accès mémoire

Les instructions load sont encodés avec le format d'instruction I et les store sont encodés avec le format S.
Détail des instructions :

- LW : load un entier 32 bits depuis la mémoire dans rd
- LH : load 16 bits depuis la mémoire en puis le signe étend sur 32 bits et le stocke dans rd

- LHU : load 16 bits mais ne signe étends pas, il zéro étend, puis le stocke dans rd
- LB : load 8 bits et signe étends
- LBU : load 8 bits et zéro étends
- SW : idem que LW mais store
- SH : idem que LH mais store
- SB : idem que LB mais store

Instruction	funct3	opcode
LW	010	0000011
LH	001	0000011
LHU	101	0000011
LB	000	0000011
LBU	100	0000011
SW	010	0100011
SH	001	0100011
SB	000	0100011

1.1 Ecall Ebreak

Ce sont des instructions systèmes qui permettent d'utiliser les fonctions du systèmes. Elles sont encodées selon l'opcode des instructions I.

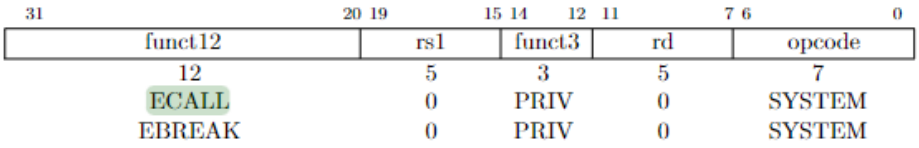
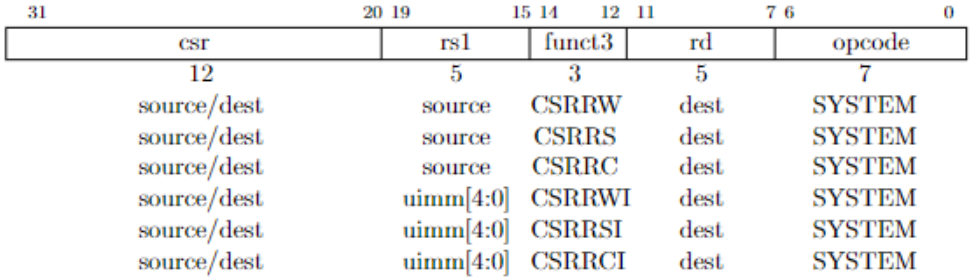


Figure 11: RISC-V systeme call

Instruction	funct12	rs1	funct3	rd	opcode
ecall	000000000000	00000	000	00000	1110011
ebreak	000000000001	00000	000	00000	1110011

1.2 CSR Instructions

Ce sont les instructions permettant d'accéder aux registres CSR, l'équivalent du CP0 en MIPS. Leur encodage est le suivant :



31 — 20	19 — 15	14 — 12	11 — 7	6 — 0
csr	rs1	funct3	rd	opcode
12	5	3	5	7
source/dest	source	CSRRW	dest	SYSTEM
source/dest	source	CSRRS	dest	SYSTEM
source/dest	source	CSRRC	dest	SYSTEM
source/dest	uimm[4:0]	CSRRWI	dest	SYSTEM
source/dest	uimm[4:0]	CSRRSI	dest	SYSTEM
source/dest	uimm[4:0]	CSRRCI	dest	SYSTEM

Figure 12: RISC-V CSR Instruction

Instruction	funct3	opcode
CSRRW	001	1110011
CSRRS	010	1110011
CSRRC	011	1110011
CSRRWI	101	1110011
CSRRSI	110	1110011
CSRRCI	111	1110011

- CSRRW : CSR read+write ou juste write, échange les valeurs du CSR et des registres.
  Si :
  - rd = 0, CSR = rs1
  - rd != 0, rd ← CSR et CSR ← rs1, , valeur zéro-étendue sur 32 bits our rd ← CSR.

- CSRRS : CSR read and Set Bits in CSR.
  Si :
  - rs1 != 0 :
  rd ← CSR, valeur zéro-étendue. rs1 est traité comme un masque qui donne la position des bits à set dans le CSR.
  Un bit à 1 dans rs1 entraine le bit correspondant dans le CSR à être mis à 1 dans le CSR (uniquement si le CSR est readable). Les autres bits ne sont pas affecté.

- rs1 = 0 :
Pas d'écriture dans les CSR.

- CSRRC : CSR read and clear Birs. Idem que CSRRC mais au lieu de set on clear le bit correspondant donc il est mis à 0.

- CSRRWI : idem que CSRRW mais immédiat à la place de rs1. Si immédiat vaut 0, on n'écrit pas. L'immédiat est zéro-étendu
- CSRRSI : idem que CSRRS mais immédiat à la place de rs1. Si immédiat vaut 0, on n'écrit pas.
- CSRRCI : idem que CSRRC mais immédiat à la place de rs1. Si immédiat vaut 0, on n'écrit pas.

1.3 Conclusion :

Instruction type	opcode
R-Type	0110011
I-Type	0010011
S-Type	0100011
B-Type	1100011
U-Type	0110111
J-Type	1101111
Système	1110011

Attention pour les accès mémoire malgré que l'encodage se face comme pour I et S, les opcodes sont différents.