Projet : Microprocesseur

Joseph Amigo, Alexandre Duplessis, François Ollivier

14 décembre 2021

Part I
ISA

(L'ISA est grandement inspirée de l'architecture RISC-V.)

1 Set d'opérations (assembleur)

Actuellement : 26 opérations → op_code se code sur 5 bits

1.1 Opérations arithmétiques et logiques

Keyword	Nom	Op_code	Détail	IC	Rq
add	Addition	00001	rd = ra + rb	R3d	
sub	Soustraction	00010	rd = ra - rb	R3d	
prod	Produit	00011	rd = ra * rb	R3d	secondaire
div	Division	00100	rd = ra // rb ; rr = ra % rb	R4d	secondaire
xor	XOR	00101	rd = ra XOR rb	R3d	
or	OR	00110	rd = ra OR rb	R3d	
and	AND	00111	rd = ra AND rb	R3d	
sll	Shift Left	01000	rd = ra << rb	R3d	
srl	Shift Rigth	01001	rd = ra >> rb	R3d	
addi	Addition imm	01010	rd = ra + imm	R2Id	
xori	Soustraction imm	01011	rd = ra XOR imm	R2Id	
ori	OR imm	01100	rd = ra OR imm	R2Id	
andi	AND imm	01101	rd = ra AND imm	R2Id	
slli	Shift Left imm	01110	rd = ra << imm	R2Id	
srli	Shift Right imm	01111	rd = ra >> imm	R2Id	
not	NOT	10000	rd = NOT ra	R2d	

*IC = Instruction Code

1.2 Opérations sur la mémoire

Keyword	Nom	Op_code	Détail	IC	Rq
load	Load	10001	rd = RAM[ra+imm]	R2Id	
limm	Load imm	10010	rd = imm	RId	
store	Store	10011	RAM[rd+imm] = ra	R2Id	
move	Move	10100	rd = ra	R2d	

1.3 Opérations de branchement

Keyword	Nom	Op_code	Détail	IC	Rq
beq	Branch ==	10101	if (rd == ra) : PC += imm	R2Id	
bne	Branch !=	10110	if (rd$\neq$ ra) : PC += imm	R2Id	
blt	Branch <	10111	if (rd < ra) : PC += imm	R2Id	
bge	Branch $\leqslant$	11000	if (rd $\leqslant$ ra) : PC += imm	R2Id	
jal	Jump and link	11001	rd = PC + 4 ; PC += imm	RId	
jalr	Jump and link reg	11010	rd = PC + 4 ; PC = ra + imm	R2Id	

*PC = Program Counter

2 Set d'instructions

31 25	25 21	21 17	17 13	13 9	9 5	5 1	Code	Exemple
op_code	rd	ra	rb	rr	NU		R4d	Div
op_code	rd	ra	rb	NU			R3d	Add
op_code	rd	ra	NU				R2d	Not
op_code	rd	ra	imm			NU	R2Id	Add imm
op_code	rd	NU	imm			NU	RId	Load imm

*NU = Non Utilisé

*imm = immidiate value

3 Architecture

3.1 Registres

Choix : 16 registres de 32 bits.

$\rightarrow$ adresse se code sur 4 bits

3.2 RAM

A priori on gère une seule RAM, et les entiers sont codés sur 32 bits. On n'accepte qu'un accès par cycle.

3.3 ROM

La ROM stocke les instructions. L'utilisateur n'y a pas accès (on a déjà un accès par cycle par le processeur). Le PC donne l'adresse de l'instruction à exécuter.

4 À ajouter ?

- bit d'état ?

Part II
Implémentation

1 Processeur

Nous avons décidé d'écrire le processeur en Python. On utilisera donc le compilateur Carotte au lieu de Minijazz.

2 Simulateur de netlist

Notre simulateur actuel est basique et écrit en Ocaml. Nous avons prévu de l'abandonner au profit d'un compilateur Netlist $\rightarrow$ C++ pour des questions d'optimisation.

3 Horloge

...

4 Divers...

- Il est prévu que nous codions notre propre assembleur.
- Il faut aussi s'occuper de l'interface de sortie

References

[Référence RISC-V] The RISC-V Instruction Set Manual, Andrew Waterman, Krste Asanovic (https://riscv.org/wp-content/uploads/2019/12/riscv-spec-20191213.pdf)

[Sujet du projet] https://github.com/hbens/sysnum-2021