

# Spécification processeur

## Projet Système Digital 2013

PAR A.AUVOLAT, E.ENGUEHARD, J.LAURENT

Nous proposons ici une spécification pour un processeur minimaliste 16 bit RISC.

### Registres

La machine dispose de 8 registres « généraux », tous de taille 16 bits :

- 0. Registre Z ou zéro (valant tout le temps 0...)
- 1. Registre A
- 2. Registre B
- 3. Registre C
- 4. Registre D
- 5. Registre E, écrit par certaines instructions (multiplication, division...) et utilisé préférentiellement comme registre temporaire pour certaines instructions composées
- 6. Registre F, ou RA (return adresse), écrit par l'instruction `jal`
- 7. Registre G, ou SP (stack pointer), utilisé par les instructions `pop` et `push`

Ces 8 registres sont tous manipulables par les instructions prenant un registre comme argument. De plus, le processeur dispose d'un registre manipulable uniquement par certaines instructions spécifiques, le registre PC (program counter : contient l'adresse de l'instruction courante).

Les numéros de registres sont donc codés sur 3 bits.

### Mémoire

La mémoire est adressée sur 16 bits et les mots mémoire font 8 bits, il y a donc 64ko disponibles.

Le CPU est little-endian (le mot 0x1234 est codé 34 puis 12)

On définit plusieurs zones de mémoire :

0x0000 - 0x3FFF	ROM pour programme utilisateur
0x4000 - 0x7FFF	MMIO (seuls quelques octets sont utilisés)
0x8000 - 0xFFFF	RAM pour programme utilisateur

**Tableau 1.** Memory map

Les bits suivants sont utilisés pour la MMIO :

byte at 0x4000	Clock ticker (incremented every tick, reset on read)
byte at 0x4100	Serial input (set when byte received, reset on read)
byte at 0x4102	Serial output (sends byte immediately on write)

**Tableau 2.** Addresses MMIO

## Jeu d'instruction

Les instructions sont codées sur 16 bits. Les tableaux suivants montrent les instructions avec les bits de poids forts à gauche et les bits de poids faibles à droite (convention contraire à celle exprimée dans le README).

### Types d'instructions

	5 bits	3 bits	8 bits
<b>Format de base</b>	<i>I</i>	<i>R</i>	...
	poids forts		poids faibles

	5 bits	3 bits	3 bits	3 bits	2 bits
<b>Format <i>R</i></b>	<i>I</i>	<i>R</i>	<i>R<sub>A</sub></i>	<i>R<sub>B</sub></i>	<i>f</i>

	5 bits	3 bits	8 bits
<b>Format <i>I</i></b>	<i>I</i>	<i>R</i>	<i>d</i>

	5 bits	3 bits	3 bits	5 bits
<b>Format <i>K</i></b>	<i>I</i>	<i>R</i>	<i>R'</i>	<i>d</i>

	5 bits	11 bits
<b>Format <i>J</i></b>	<i>I</i>	<i>d</i>

### Tableau d'instructions

Certain noms d'instructions sont en *italique*, il s'agit de signifier qu'il s'agit d'un alias (optionnel) pour une autre instruction.

<i>I</i>	format	<i>f</i>	description	action	valeurs signés ?
00000	R	0	add	$R \leftarrow R_A + R_B$	signé
		1	sub	$R \leftarrow R_A - R_B$	signé
		2	mul	$R \leftarrow \text{lo}(R_A \times R_B)$	signé
				$E \leftarrow \text{hi}(R_A \times R_B)$ si $E \neq R$	
		3	div	$R \leftarrow q(R_A, R_B)$	signé
				$E \leftarrow r(R_A, R_B)$ si $E \neq R$	
00001	R	0	addu	idem add	non signé
		1	subu	idem sub	non signé
		2	mulu	idem mul	non signé
		3	divu	idem div	non signé
00010	R	0	or	$R \leftarrow (R_A \vee R_B)$	
		1	and	$R \leftarrow (R_A \wedge R_B)$	
		2	xor	$R \leftarrow (R_A \oplus R_B)$	
		3	nor	$R \leftarrow \text{not}(R_A \vee R_B)$	
00011	R	0	lsl	$R \leftarrow (R_A \ll R_B)$	
		1	lsl		
		2	lsr	$R \leftarrow (R_A \gg R_B)$ (logical)	(non signé)
		3	asr	$R \leftarrow (R_A \gg R_B)$ (arith)	(signé)
00100	R	0	se		
		1	sne		
		2	se	$R \leftarrow (R_A = R_B ? 1 : 0)$	
		3	sne	$R \leftarrow (R_A \neq R_B ? 1 : 0)$	
00101	R	0	slt	$R \leftarrow (R_A < R_B ? 1 : 0)$	signé
		1	sle	$R \leftarrow (R_A \leq R_B ? 1 : 0)$	signé
		2	sltu	$R \leftarrow (R_A < R_B ? 1 : 0)$	non signé
		3	sleu	$R \leftarrow (R_A \leq R_B ? 1 : 0)$	non signé
00110	I		incrl	$R \leftarrow (R + d)$	<i>d</i> signé
00111	I		shl	$R \leftarrow (R \ll d)$	<i>d</i> signé
01000	J		j	$\text{PC} \leftarrow \text{PC} + d$	
01001	J		jal	$F \leftarrow (\text{PC} + 2); \text{PC} \leftarrow \text{PC} + d$	
01010	R	0	jr	$\text{PC} \leftarrow R$	
		1	jalr	$F \leftarrow (\text{PC} + 2); \text{PC} \leftarrow R$	
		2	jer	if $R_A = R_B$ then $\text{PC} \leftarrow R$	
		3	jner	if $R_A \neq R_B$ then $\text{PC} \leftarrow R$	
01011	R	0	jltr	if $R_A < R_B$ then $\text{PC} \leftarrow R$	signé
		1	jler	if $R_A \leq R_B$ then $\text{PC} \leftarrow R$	signé
		2	jltru	if $R_A < R_B$ then $\text{PC} \leftarrow R$	non signé
		3	jltru	if $R_A \leq R_B$ then $\text{PC} \leftarrow R$	non signé
01100	J		lra	$E \leftarrow \text{PC} + d$	<i>d</i> signé
01101			nop		
01110			nop		
01111			hlt	halt microprocessor (infinite loop)	
10000	K		lw	$R \leftarrow \text{mem}(R' + d)$ (16 bits)	
10001	K		sw	$\text{mem}(R' + d) \leftarrow R$ (16 bits)	
10010	K		lb	$R_{\text{lo}} \leftarrow \text{mem}(R' + d); R_{\text{hi}} \leftarrow 0$ (8 bits)	
10011	K		sb	$\text{mem}(R' + d) \leftarrow R_{\text{lo}}$ (8 bits)	
10100	R	*	lwr	$R \leftarrow \text{mem}(R_A + R_B)$ (16 bits)	
10101	R	*	swr	$\text{mem}(R_A + R_B) \leftarrow R$ (16 bits)	
10110	R	*	lbr	$R_{\text{lo}} \leftarrow \text{mem}(R_A + R_B); R_{\text{hi}} \leftarrow 0$ (8 bits)	
10111	R	*	sbr	$\text{mem}(R_A + R_B) \leftarrow R_{\text{lo}}$ (8 bits)	
11000	I		lil	$R_{\text{lo}} \leftarrow d$	
11001	I		lilz	$R_{\text{lo}} \leftarrow d; R_{\text{hi}} \leftarrow 0$	( <i>d</i> non signé)
11010	I		liu	$R_{\text{hi}} \leftarrow d$	
11011	I		liuz	$R_{\text{hi}} \leftarrow d; R_{\text{lo}} \leftarrow 0$	
11100	I		lie	$R \leftarrow \text{sign\_extend}_8^{16}(d)$	( <i>d</i> signé)
11101			nop		
11110			nop		
11111			nop	3 $\emptyset$	

**Tableau 3.** Instructions reconnues par le microprocesseur

L'assembleur propose également quelques instructions « étendues » permettant de faciliter la programmation :

Nom	Action	Code assembleur de base équivalent
push $R$	$G \leftarrow G - 2; \text{mem}(G) \leftarrow R$	incv $G, -2$ sw $R, G$
pop $R$	$R \leftarrow \text{mem}(G); G \leftarrow G + 2$	lw $R, G$ incv $G, 2$
move $R, R_A$	$R \leftarrow R_A$	add $R, R_A, Z$
addi, subi, ...	$R \leftarrow R_A + d$	(utilise E comme registre temporaire)
not $R, R_A$	$R \leftarrow \text{not } R_A$	nor $R, R_A, Z$
jz $R, \text{addr}$	if $R = 0$ then PC $\leftarrow \text{addr}$	lil E, lo(addr) ; liu E, hi(addr) OU lilz E, addr jer R, E, Z
jnz $R, \text{addr}$	if $R \neq 0$ then PC $\leftarrow \text{addr}$	lil E, lo(addr) ; liu E, hi(addr) OU lilz E, addr jner R, E, Z

**Tableau 4.** Instructions supplémentaires (produites par l'assembleur)

