Spécification processeur

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

Registres

La machine dispose de 8 registres « généraux » :

- 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`

De plus, le processeur dispose d'un registre non manipulable, le registre `PC` (program counter).

Les numéros de registres sont donc codés sur 3 bits.

Mémoire

La mémoire est adressée sur 16 bits, il y a donc 64ko disponnibles.

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

Modèle simple

On définit plusieurs zones de mémoire :

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

Tableau 1. Memory map

Modèle avec affichage bitmapé

De moins en moins de chances d'être implémenté... mais ça n'a rien d'impossible.

On définit plusieurs zones de mémoire :

0x0000 - 0x3FFF	ROM pour programme utilisateur
0x4000 - 0x6FFF	VGA Framebuffer (noir et blanc, 336x288)
0x7000 - 0x77FF	ROM pour police d'écriture
0x7800 - 0x7FFF	MMIO (seuls quelques octets seront utilisés)
0x8000 - 0xFFFF	RAM pour programme utilisateur

Tableau 2. Memory map

Les 0x3000 (12288) octets de mémoire pour le VGA correspondent à un affichage bitmapé 336x288 noir et blanc (un octet représente 8 pixels), ce qui fait avec une police d'écriture 8x8 un affichage texte possible en 42x36.

Les 0x0800 (2048) octets de RAM pour la fonte suffisent à définir 256 caractères en résolution 8x8 (donc 8 octets par caractère).

Sur les 0x8000 octets alloués pour la MMIO, on en aura un pour l'entrée série, un pour la sortie série, un pour l'horloge et c'est tout.

Le reste est auto-explicite.

Jeu d'instruction

Les instructions sont codées sur 16 bits.

Types d'instructions

Format de base

5 bits	3 bits	8 bits
I	R	...

Format *R*

5 bits	3 bits	3 bits	3 bits	2 bits
I	R	R_A	R_B	f

Format *I*

5 bits	3 bits	8 bits
I	R	d

Format K

5 bits	3 bits	3 bits	5 bits
I	R	R'	d

Format *J*

5 bits	11 bits
I	d

Tableau d'instructions

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

I	format	f	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	$R \leftarrow (R_A \leqslant R_B ? 1 : 0)$	signé
		2	sltu	$R \leftarrow (R_A < R_B ? 1 : 0)$	non signé
		3	sleu	$R \leftarrow (R_A \leqslant R_B ? 1 : 0)$	non signé
00110	I		incri	$R \leftarrow (R + d)$	d signé
00111	I		shi	$R \leftarrow (R \ll d)$	d 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 $R_A \leqslant R_B$ then $\text{PC} \leftarrow R$	signé
		2	jltru	if $R_A < R_B$ then $\text{PC} \leftarrow R$	non signé
		3	jleru	if $R_A \leqslant R_B$ then $\text{PC} \leftarrow R$	non signé
01100			*nop*		
01101			*nop*		
01110			*nop*		
01111			*nop*		
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	$R \leftarrow \text{mem}(R' + d)$ (8 bits)	
10011	K		sb	$\text{mem}(R' + d) \leftarrow R$ (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	$R \leftarrow \text{mem}(R_A + R_B)$ (8 bits)	
10111	R	*	sbr	$\text{mem}(R_A + R_B) \leftarrow R$ (8 bits)	
11000	I		lil	$R_{\text{lo}} \leftarrow d$	
11001	I		lilz	$R_{\text{lo}} \leftarrow d\,; R_{\text{hi}} \leftarrow 0$	
11010	I		liu	$R_{\text{hi}} \leftarrow d$	
11011	I		liuz	$R_{\text{hi}} \leftarrow d\,; R_{\text{lo}} \leftarrow 0$	
11100			*nop*		
11101			*nop*		
11110			*nop*		
11111			nop		

Tableau 3. Instructions reconnues par le microproceseur

Nom	**Action**	**Code assembleur de base équivalent**
push R	$G \leftarrow G-2; \text{mem}(G) \leftarrow R$	incri $G, -2$ sw R, G
pop R	$R \leftarrow \text{mem}(G)\,; G \leftarrow G+2$	lw R, G incri 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, addr	if $R = 0$ then $\text{PC} \leftarrow$ addr	lil E, lo(addr) ; liu E, hi(addr) OU lilz E, addr jer R, E, Z
jnz R, addr	if $R \neq 0$ then $\text{PC} \leftarrow$ 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)