Choix relatifs à l'implémentation du processeur

Marc Ducret Florentin Guth Martin Ruffel
Lionel Zoubritzky

10 décembre 2016

1 Structure du système

1.1 Architecture du processeur

On choisit de stocker les mots sur 32 bits (4 octets). En effet, cela permettra d'utiliser des entiers (signés seulement, pour simplifier) de taille importante. On utilise 32 registres, ce qui, en les stockant sur 5 bits, laisse 17 bits pour l'`opcode` des instructions à 3 paramètres.

1.2 Caractéristiques de la *RAM*

On souhaiterait pouvoir effectuer de l'affichage sur un écran de 256×256 pixels avec des couleurs de la taille d'un mot (32 bits). Cela nécessite donc 256 Ko d'espace mémoire. On choisit donc une *RAM* de 512 Ko.

La *RAM* contiendra des emplacements réservés pour :

— initialiser le microprocesseur,

— contenir le programme de lancement (la montre digitale),

— afficher une *bitmap* à l'écran,

— gérer les entrées restantes : changement de mode de la montre, remise à l'heure ...

2 Opérations implémentées

2.1 Opérations arithmético-logiques

On conserve pratiquement toutes les opérations de base du *MIPS*.

Opération				Effet				
`add`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	r_2	$+$	o
`sub`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	r_2	$-$	o
`mul`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	r_2	$\times$	o
`div`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	r_2	$\div$	o
`sll`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	r_2	lsl	o
`srl`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	r_2	lsr	o
`and`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	r_2	$\wedge$	o
`or`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	r_2	$\vee$	o
`xor`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	r_2	$\oplus$	o
`slt`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	r_2	$<$	o
`sle`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	r_2	$\leq$	o
`seq`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	r_2	$=$	o
`sne`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	r_2	$\neq$	o

Table 1 – Opérations arithmético-logiques

On ajoute néanmoins les opérations suivantes, afin de faciliter les opérations sur les heures :

Opération				Effet				
`add24`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	$r_2 + o$	mod	24
`add60`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	$r_2 + o$	mod	60
`mod`	`r1,`	`r2,`	`o`	r_1	$\leftarrow$	r_2	mod	o
`mod24`	`r1,`	`r2`		r_1	$\leftarrow$	r_2	mod	24
`mod60`	`r1,`	`r2`		r_1	$\leftarrow$	r_2	mod	60

TABLE 2 – Opérations arithmético-logiques

On ajoute que les opérations `add24` et `add60` lèveront un *flag* lorsqu'un dépassement aura eu lieu.

2.2 Opérations de manipulations de données

Parmi les opérations du *MIPS*, on conserve les suivantes. On a choisi de ne pas inclure d'opération de type `load_immediate`, car on stockera dans certains registres les constantes nécessaires avant le lancement du processeur (comme 0, 24, 60...).

Opération				Effet		
`move`	`r1,`	`r2`		r_1	$\leftarrow$	r_2
`lw`	`r1,`	`o`	`(r2)`	r_1	$\leftarrow$	RAM$[r_2 + o]$
`rw`	`r1,`	`o`	`(r2)`	r_1	$\rightarrow$	RAM$[r_2 + o]$

TABLE 3 – Opérations de manipulations de données

2.3 Opérations de contrôle de l'exécution

Enfin, on dispose des opérations suivantes afin de simuler des conditions et des boucles.

Opération				Effet			
`j`	`o`			PC	$\leftarrow$	o	
`beq`	`r,`	`o,`	`a`	PC	$\leftarrow$	a	si r = o
`nop`				$\varnothing$			

TABLE 4 – Opérations de contrôle de l'exécution