BFComputer Specifications

BFComputer – abrégé BFCom, ou BFC – est un ordinateur 12-bits fonctionnant avec le langage hétéroclite Brainf*ck en guise de langage d'assembleur. Ainsi, il ne comprend que huit instructions basiques qui seront détaillé dans la prochaine partie. BUT PROJET ETC… AUTRE TRUCS A DIRE ?

Le Brainf*ck et les instructions

Le Brainf*ck est donc composé de huit instructions élémentaires permettant de manipuler une mémoire, le déroulement du programme ainsi qu'interagir avec des entrées/sorties. EXPLICATION JEU INSTRUCTIONS BF

+	Incrémente la valeur de la cellule courante
-	Décrémente la valeur de la cellule courante
>	Incrémente le curseur de la mémoire
<	Décrémente le curseur de la mémoire
[	Début d'une boucle
]	Fin d'une boucle
.	Envoie la valeur de la cellule courante à la sortie
,	Écris la valeur de l'entrée dans la cellule courante

L'architecture de l'ordinateur

L'ordinateur doit donc avoir une architecture spécifique pour interpréter ce jeu d'instruction réduit. L'ALU – Arithmetic and Logic Unit – est assez simple car il se résume à incrémenter ou décrémenter la valeur de registres. Cependant, ce qui complexifie la tâche est le système de boucle. Dans un langage d'assembleur normal, il n'existe pas de boucle à proprement parlé mais des 'sauts' avec ainsi l'adresse d'arrivée indiquée dans l'instruction. Ici le challenge a été de trouver un mécanisme permettant à l'ordinateur d'aller chercher le début ou la fin de la boucle correspondante. De plus, pour optimiser un peu le tout – bien que cela ne soit pas forcément utile étant donné que dans tous les cas l'ordinateur sera lent, son horloge fonctionnant à X MHz max – chaque instruction de boucle va être stocké dans une mémoire cache parallèle à celle utilisable pour l'utilisateur.

L'architecture peut être décomposée en deux parties : la logique de contrôle, qui permet de décoder l'instruction courante et de l'exécuter ; ainsi que logique de calcul, qui contient les mémoires, la RAM, l'ALU et les entrées/sorties. Mais avant tout, parlons de l'horloge principale de l'ordinateur.

L'horloge

L'horloge est globalement la même que celle de l'ordinateur 8-bits de Ben Eater à l'exception qu'il n'est pas possible de régler sa vitesse. Il y a donc deux modes : un mode manuel – permettant de debug – contrôlé par un bouton avec un circuit de debounce à base de timer 555 ; ainsi qu'un mode normal utilisant un timer 555 en astable à une fréquence de X MHz. De plus, il n'y a pas de signal de contrôle permettant de stopper l'horloge, cela sera fait en créant une boucle infinie dans le code.

La logique de calcul
Les registres

L'ordinateur est constitué de quatre registres :

Nom	Sigle	Utilité
Program Counter	PC	Registre contenant le pointeur permettant de garder une trace de l'instruction qui s'éxécute.
Memory Address Register	MAR	Registre contenant le curseur/pointeur de la mémoire.
Loop Address Register	LAR	Registre contenant le curseur de la mémoire servant de cache aux emplacements des boucles.
Loop Counter	LPC	Permet de compter le nombre de boucle rencontré lorsque l'on cherche l'instruction opposée.

Le LAR et le MAR sont reliés via un multiplexer au bus d'adresse de la mémoire RAM lui même relié au bus de données.
Tous les registres de cet ordinateur sont basés sur un circuit intégré de type 74LS169 avec trois différents signaux de contrôles :

- CO : permet d'activer le comptage, décrémentation par défaut
- UP : permet d'incrémenter lors d'un comptage
- LO/RESET : permet de mettre une valeur dans le registre. Cette valeur étant 0 lorsque le signal est dénommé RESET

L'ALU

L'ALU n'est en réalité rien d'autre que un registre relié au bus de données et permet d'incrémenter ou décrémenter sa valeur interne. À des fins d'optimisation, la valeur en RAM y est stocké que lorsque nous cherchons à changer sa valeur. De même, sa valeur est retournée en RAM seulement si il y a eu modification et que lorsque l'on modifie le curseur mémoire .

La RAM

La RAM a une organisation de 8k x 12. Cependant, seuls 4k sont adressables, en effet, la mémoire est coupée en deux entre la mémoire utilisable et la mémoire de cache pour l'optimisation des boucles.
Actuellement, le circuit intégré retenu est le AS6C6264-55PIN, ayant une organisation de 8k x 8 donc devant être doublé. Ses signaux de contrôles sont les suivants :

- OE : permet de mettre la valeur de la cellule actuelle sur le bus de données
- WE : permet de stocker la valeur du bus de données dans la cellule actuelle

De plus, il existe deux registres d'adressage différent : un utilisé pour le curseur de l'utilisateur (MAR) et un utilisé pour le cache des boucles (LAR). Les deux permettent d'adresser la RAM

simultanément via un multiplexer. Le MAR est relié à l'adresse 0x00 du multiplexer et le LAR à l'adresse 0x01.

Les entrées/sorties

Plusieurs périphériques peuvent faire office d'entrée/sorties, cependant, les signaux de contrôles universels seront :

- Pour la sortie :
  - PA : transmet la valeur de l'ALU
  - PB : transmet la valeur du bus de données
- Pour l'entrée :
  - Signal correspondant à la sélection via le multiplexer du bus, à l'adresse 0x02

Le bus de données

Le bus de données est relié à la RAM, l'ALU, le PC ainsi que les entrées/sorties. Il est par défaut mis à zéro et est ainsi utilisé pour initialiser les registres et la RAM à cette valeur.
Il y a deux seuls moyens d'un mettre une valeur : soit par la sortie de la RAM soit par un multiplexer où sont reliés d'autres éléments de l'ordinateur à différentes adresses :

- 0x00 : ALU
- 0x01 : PC
- 0x02 : IO

Les Zero Checkers

Les ''Zero Checkers'' permettent de vérifier si une valeur est nulle ou pas. L'ordinateur en possède deux : un pour la valeur du bus, son résultat étant enregistré dans un registre de flag quand nécessaire et un deuxième pour la valeur du LPC, son résultat étant directement utilisé pour déterminer la micro-instruction.

La logique de contrôle

La logique de contrôle est en soit assez simple : elle réunit tous les éléments nécessaire à la détermination des opérations que la logique de calcul doit effectuer. Nous avons donc pour cela une ROM principale avec 8192 cellules de 25 bits. 8192 cellules correspondent à 13 bits et plus précisément quatre éléments résumé dans le tableau ci dessous :

Element		Bit Index
Program ROM		#0
		#1
		#2
Phase		#3
		#4
Zero Checker	LFF : Loop Found Flag	#5
Flag Register	SLF : Start Loop Flag	#6

	ELF : End Loop Flag	#7
	ACF : ALU Changed Flag	#8
	BZF : Bus Zero Hold	#9
	RF : Reset Flag	#10

Voici une brève explication de chaque élément :

Program ROM
La ROM qui contient le programme à exécuter. Son contenu peut être changé via le programmateur de ROM dont une partie de ce projet est aussi dédiée. Les instructions défilent via le PC.

Phase
Registre de 2-bit qui bascule entre zéro et potentiellement quatre pour les instructions nécessitant plusieurs cycles d'horloge, cependant, seulement trois cycles sont utiles pour éxécuter n'importe quel instruction.

Zero Checker
Permet d'indiquer à l'ordinateur quand le LPC est à zéro.

Flag Register
Registre de flag indiquant certains états interne de l'ordinateur.
Voici une description de chaque flag :

- SLF : permet d'indiquer à l'ordinateur que nous sommes à la recherche du début d'une boucle, soit à une instruction BF '['
- ELF : inversement, permet d'indiquer que nous cherchons la fin d'une boucle, soit ']'
- ACF : permet d'indiquer quand la valeur de l'ALU a été changé et donc s'il est nécessaire de mettre à jour la valeur de l'ALU ou de la cellule en RAM
- BZF : permet d'indiquer à l'ordinateur si la valeur du bus est à zéro. Nécessite un signal de contrôle car ce flag est en réalité stocké dans un registre
- RF : permet d'indiquer à l'ordinateur que nous sommes en phase de (ré)initialisation. Il est activé lors de l'appuie d'un bouton et est réintialisé lorsque la valeur du MAR est à sa valeur max 0xFFF

Les séquences d'instruction

Fetch Cycle

Le « Fetch Cycle » est l'opération qui consiste à récupérer l'instruction suivante dans le but de la décoder puis l'exécuter. Elle est donc réalisée après chaque instruction et consiste à réinitialiser la phase, et incrémenter le PC.

La réinitialisation de l'ordinateur

La réinitialisation de l'ordinateur (qui est aussi son initialisation) consiste à mettre tous les registres ainsi que le contenu de la RAM à zéro. Cela consiste à trois étapes distinctes :

1. Initialiser PC-MAR-Phase-Flag-LPC à 0 via LOAD => à l'appuie du bouton de reset.
2. PC → LAR & Bus → RAM

3. Incrémenter MAR jusqu'à la valeur max (4095) où le RF est réinitialisé

La gestion des boucles

sdfdsqgfdsf

NB: Limitation = ne peut pas avoir de boucle dès la première instruction : workaround : mettre >< ou <> juste avant