Projet
Architecture et Modèle de calcul

Objectif :
Écrire en C un simulateur de processeur 20-bits.

Impératifs :
Il s'agit d'un travail **personnel**. Tout plagiat sera sanctionné. Les fichiers sources devront être commentés. Votre projet devra utiliser les outils de création de projet (CMAKE) et devra comporter un README explicatif ainsi que des programmes de validation et test.

Détails :
Ce projet se décompose en 2 parties consécutives qui devront être réalisée dans l'ordre.

1) Architecture simple non-pipelinée basée sur IAS (à rendre pour le **1er mars**)
2) Architecture pipelinée à 5 étages (à rendre pour le **1er avril**)

1. Architecture simple non-pipelinée basée sur IAS

Pour cette partie du projet vous devrez réaliser un simulateur de processeur non-pipelinée basée sur l'architecture IAS suivante.

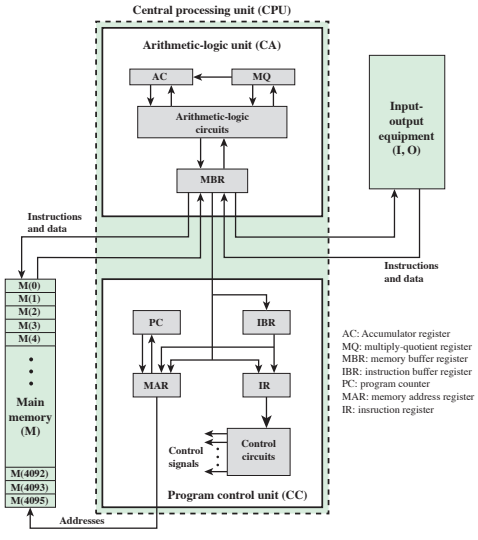

Figure 1.6 IAS Structure

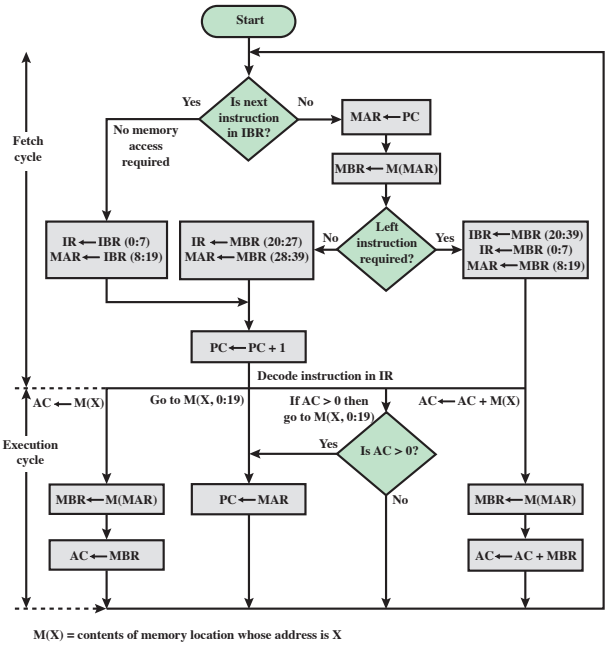

Figure 1.8 Partial Flowchart of IAS Operation

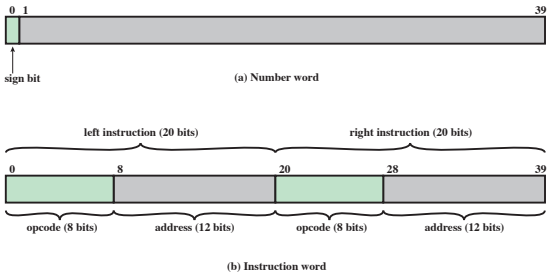

Figure 1.7 IAS Memory Formats

L'IAS fonctionne en exécutant de manière répétitive un cycle d'instructions, comme indiqué dans la figure 1.8. Chaque cycle d'instruction se compose de deux sous-cycles. Pendant le cycle de fetch, l'opcode de l'instruction suivante est chargé dans l'IR et la partie d'adresse est chargée dans le MAR. Cette instruction peut être tirée de l'IBR, ou elle peut être obtenue à partir de la mémoire en chargeant un mot dans le MBR, puis dans l'IBR, IR et MAR.

Ces opérations sont contrôlées par des circuits électroniques conduisant à l'utilisation de chemins de données. Pour simplifier l'électronique, il n'y a qu'un seul registre utilisé pour spécifier l'adresse en mémoire pour une lecture ou une écriture et uniquement un registre utilisé pour la source ou la destination.

Une fois que l'opcode est dans l'IR, le cycle d'exécution est exécuté. Le circuit de contrôle interprète l'opcode et exécute l'instruction en envoyant les signaux de commande pour provoquer le déplacement des données ou l'exécution d'une opération par l'ALU.

Le jeu d'instruction de cette architecture est le suivant :

Instruction Type	Opcode	Symbolic Representation	Description
Data transfer	00001010	LOAD MQ	Transfer contents of register MQ to the accumulator AC
	00001001	LOAD MQ,M(X)	Transfer contents of memory location X to MQ
	00100001	STOR M(X)	Transfer contents of accumulator to memory location X
	00000001	LOAD M(X)	Transfer M(X) to the accumulator
	00000010	LOAD –M(X)	Transfer –M(X) to the accumulator
	00000011	LOAD |M(X)|	Transfer absolute value of M(X) to the accumulator
	00000100	LOAD –|M(X)|	Transfer –|M(X)| to the accumulator
Unconditional branch	00001101	JUMP M(X,0:19)	Take next instruction from left half of M(X)
	00001110	JUMP M(X,20:39)	Take next instruction from right half of M(X)
Conditional branch	00001111	JUMP+ M(X,0:19)	If number in the accumulator is nonnegative, take next instruction from left half of M(X)
	00010000	JUMP+ M(X,20:39)	If number in the accumulator is nonnegative, take next instruction from right half of M(X)
Arithmetic	00000101	ADD M(X)	Add M(X) to AC; put the result in AC
	00000111	ADD |M(X)|	Add |M(X)| to AC; put the result in AC
	00000110	SUB M(X)	Subtract M(X) from AC; put the result in AC
	00001000	SUB |M(X)|	Subtract |M(X)| from AC; put the remainder in AC
	00001011	MUL M(X)	Multiply M(X) by MQ; put most significant bits of result in AC, put least significant bits in MQ
	00001100	DIV M(X)	Divide AC by M(X); put the quotient in MQ and the remainder in AC
	00010100	LSH	Multiply accumulator by 2; i.e., shift left one bit position

	00010101	RSH	Divide accumulator by 2; i.e., shift right one position
Address modify	00010010	STOR M(X,8:19)	Replace left address field at M(X) by 12 rightmost bits of AC
	00010011	STOR M(X,28:39)	Replace right address field at M(X) by 12 rightmost bits of AC

L'architecture IAS a un total de 21 instructions, présentées dans le tableau précédent. Elles peuvent être regroupées comme suit :

- **Transfert de données** : déplace les données entre la mémoire et les registres ALU ou entre deux registres ALU.
- **Branchement inconditionnel** : Normalement, l'unité de contrôle exécute les instructions en séquence depuis la mémoire. Cette séquence peut être modifiée par une instruction de branchement, ce qui facilite les opérations répétitives.
- **Branchement conditionnel** : la branche peut être rendue dépendante d'une condition.
- **Arithmétique** : opérations effectuées par l'ALU.
- **Modification de l'adresse** : permet de calculer les adresses dans l'ALU, puis de les insérer dans des instructions stockées en mémoire. Cela donne de la flexibilité aux programmes.

Le tableau précédent présente les instructions sous une forme symbolique facile à lire. En pratique chaque instruction doit être conforme au format de la figure 1.7. La partie opcode (8 premiers bits) spécifie laquelle des 21 instructions doit être exécutée. La partie adresse (12 bits restants) spécifie lequel des 4096 emplacements mémoire doit être sélectionné lors de l'exécution.

Notez que certaines opérations nécessitent plusieurs cycles pour être exécuter par l'unité de contrôle. Par exemple, la multiplication a besoin de 39 sous-opérations, une pour chaque bit à l'exception du bit de signe.

Travail demandé :
Vous devrez donc écrire un simulateur de processeur IAS en C, prenant en entrée un fichier binaire comprenant du code IAS et capable de le décoder et de l'exécuter. Les contraintes sont les suivantes :

1) L'exécution devra permettre d'observer l'état de chaque élément du processeur IAS, à chaque étape. Pour cela, vous devrez écrire une fonction par élément.
2) Le transfert d'information entre chaque fonction devra se faire exclusivement par passage d'argument.
3) Vous devrez proposer un mode « debug » activé par défaut, permettant d'observer l'état du processeur à chaque instant

Pour vous aider, vous réaliserez un « compilateur » prenant en entrée un fichier en assembleur IAS et fournissant en sortie un binaire.
Le travail demandé se rapproche de celui disponible sur le web à :
https://www.ic.unicamp.br/~edson/disciplinas/mc404/2017-2s/abef/IAS-sim/
(L'interface graphique n'est pas nécessaire)