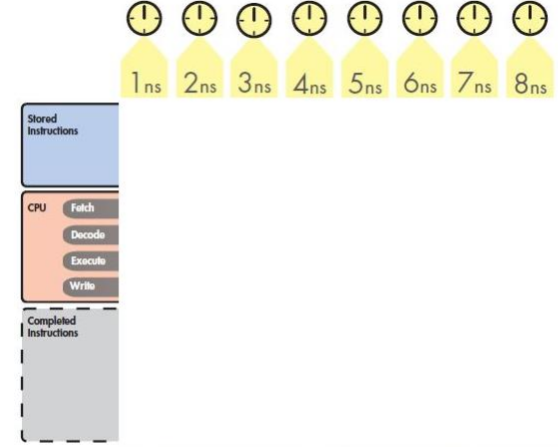
1. Complétez le schéma de ce pipeline pour un ensemble de 4 instructions, chaque étage du pipeline prenant 1 ns (la partie Stored Instructions n'est pas prise en compte pour la correction). 

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Stored | Y | Z | A |  |  |  |  |  |
| Fetch | X | Y | Z | A |  |  |  |  |
| Decode |  | X | Y | Z | A |  |  |  |
| Execute |  |  | X | Y | Z | A |  |  |
| Write |  |  |  | X | Y | Z | A |  |
| Completed |  |  |  |  | X | Y | Z | A |
|  |  |  |  |  |  | X | Y | Z |

a. Donnez le completion rate théorique maximum de ce pipeline.

4 Instructions / 8 ns -> 0.5ins/ns

b. Donnez le completion rate réel moyen de ce pipeline après 20ns si le celui-ci n’était pas préalablement chargé.

15 Instructions / 20 ns -> 0,75ins/ns

c. Donnez le program execution time d’un programme comprenant 15 instructions (sur base du completion rate théorique calculé ci-dessus).

15 Instructions / Completion rate = 0.75 -> 20 ns

1. Illustrez graphiquement le phénomène de décrochage du pipeline.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Stored | Y | Z | A | A |  |  |  |  |  |
| Fetch | X | Y | Z | Z | A |  |  |  |  |
| Decode |  | X | Y | Y | Z | A |  |  |  |
| Execute |  |  | X | ------ | Y | Z | A |  |  |
| Write |  |  |  | X |  | Y | Z | A |  |
| Completed |  |  |  |  | X |  | Y | Z | A |
|  |  |  |  |  |  | X |  | Y | Z |

Citez une des causes principales de ces décrochages :

Accès à la mémoire centrale trop lent

1. Citez les 2 limites du pipeline (pourquoi ne pas avoir des profondeurs énormes ?). Expliquez brièvement.

… Si une étape est plus longue, elle ralentira toutes les instructions

… Une architecture à pipeline est coûteuse et représente donc une limite

1. Expliquez les commandes suivantes (soyez précis, utilisez les termes corrects) :
   1. Load #13, D

Charger le contenu de l’adresse du registre 13 dans le registre D

* 1. Load #C, B

Charger le contenu de l’adresse contenue dans C dans le registre B

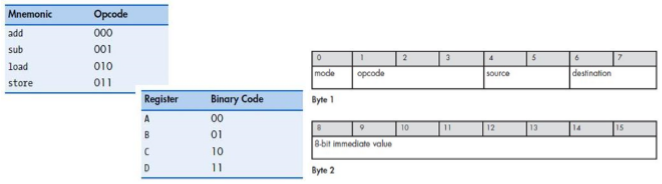
* 1. Store B, #(D+17)

Placer le contenu du registre B dans l’adresse contenue dans le registre D + 17 ( X )

* 1. Add B, C, C

Additionner le contenu du registre B et du registre C et écrire le résultat dans le registre C

1. Soit le mapping et le format d’instructions suivant :



* 1. Traduire en langage machine l’instruction suivante : **sub** *C*, **A**, A

0 **001** *10* **00** 00 000000

* 1. Traduire en langage assembleur l’instruction suivante : 1000110100110101

1 000 11 01 0011.0101

* Add D, 53, B

1. Définir en quelques mots :
   1. Un registre :

Espace de stockage dans le processeur

* 1. Le program counter :

Indique l’adresse de la prochaine instruction à charger

* 1. Le programming model :

Modèle qui décrit les ressources disponibles

* 1. Un bus :

Fil où navigue les données

* 1. L’instruction register :

Décode l’instruction

* 1. Le PSW :

Processus Status Word, saute vers une autre partie du programme si une condition est remplie

* 1. L’assembleur :

Traduit le programme en langage machine vers le langage humain

1. Comment les architectes systèmes ont-ils rendu plus rapide la lecture des données par l’ALU ?

En utilisant les registres internes au CPU

1. Citer 4 catégories d’instructions.

* Arithmétiques (add, sub, etc…)
* Accès à la mémoire (load)
* Logique (and, or, not)
* Embranchement (se déplacer dans le flux)

1. Quel avantage l’utilisation d’adresse relative offre au programmeur ? Expliquer brièvement.

Le programmeur n’avait plus besoin de connaître l’emplacement exact des données en mémoire, il devait juste connaître le registre dans lequel l’OS place l’adresse de base du segment.

1. Pourquoi était-il nécessaire d’encoder les programmes en format binaire dans les premiers ordinateurs ?

Il n’y avait pas assez de ressources sur les premiers ordinateurs pour mettre en place une interface utilisateur. La grande majorité des ressources étaient données au programme

1. Expliquer la boucle Fetch-Decode-Execute.

Fetch lit l’instruction et la charge, Decode la décode et Execute l’exécute. C’est une sorte de read add write.

1. Qu’est-ce que le PSW ? A quel type d’instruction est-il principalement utile ?

Processus Status Word, saute vers une autre partie du programme si une condition est remplie. Il sert aux instructions unconditional (jump).