4- Chemin d'exécution

Quelle architecture matérielle faut-il pour implémenter cette ISA ?

- Principe expliqué pour l'ISA x86 simplifiée
- HYPOTHESE: toutes les instructions prennent un cycle, les instructions font toutes la même taille
- Instructions étudiées

```
mov, add, sub, or, and, jeq, jne
```

Chemin d'exécution:

- Description de toutes les unités matérielles et leur connection permettant l'exécution de programmes assembleurs

4-a Décodage

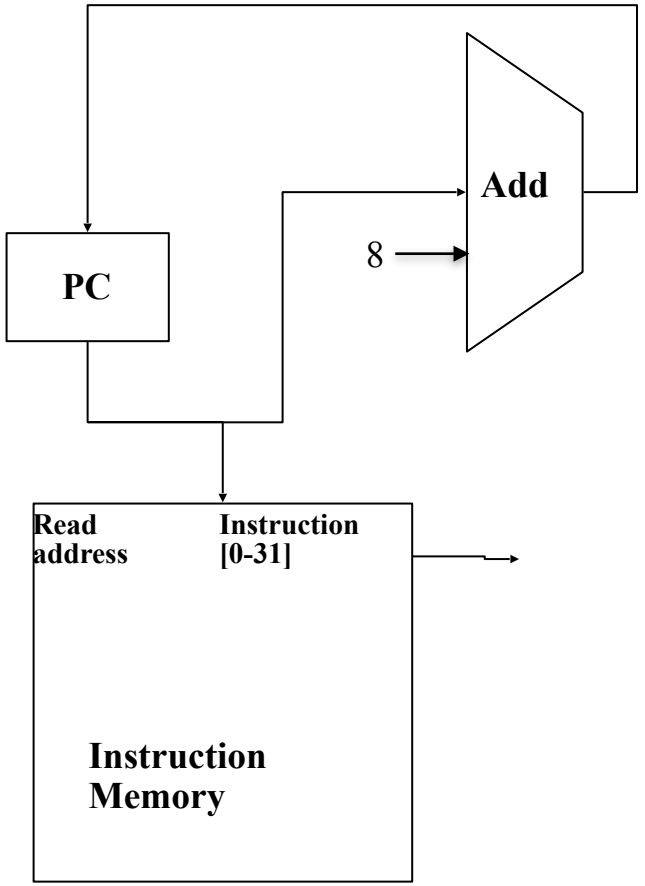
Décodage en boucle des instructions

- le compteur ordinal (program counter) contient l'adresse de la prochaine instruction
- 1 instruction x86 = 32 bits
- continuellement mis à jour
- on suppose que la mémoire des instructions est différentes de celle des données

4-a Décodage

Décodage en boucle des instructions

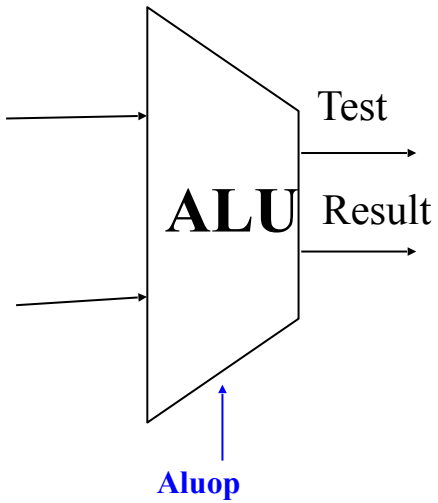
- le compteur ordinal (program counter) contient l'adresse de la prochaine instruction
- 1 instruction x86 = 32 bits
- continuellement mis à jour
- on suppose que la mémoire des instructions est différentes de celle des données



4-b Instructions arithmétiques

ALU: unité arithmétique et logique

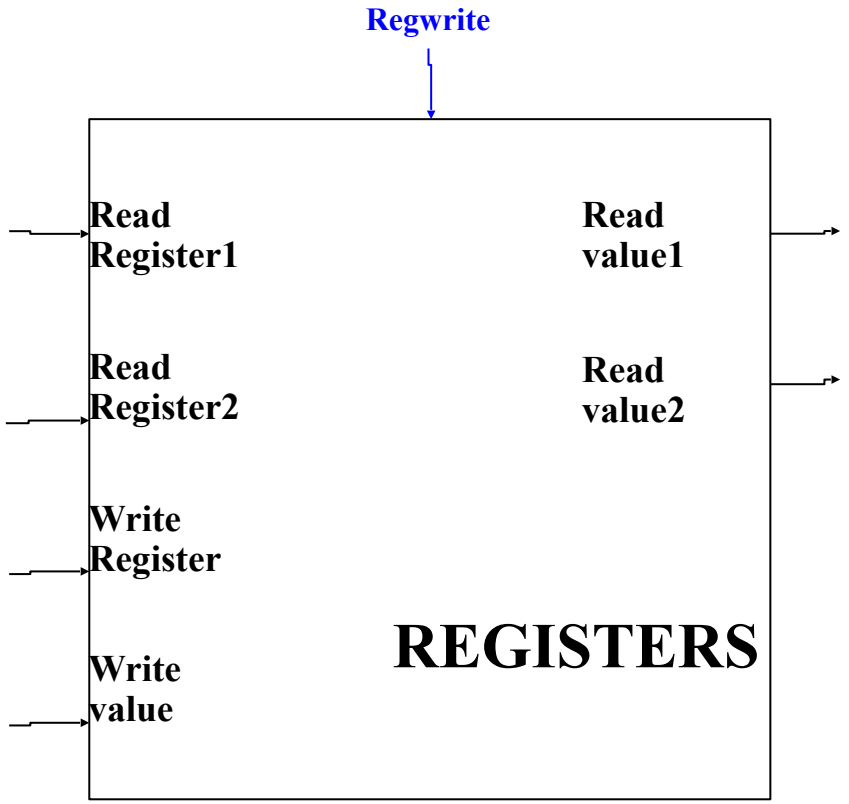
- prend deux valeurs
- retourne la valeur résultat de l'opération, controlée par la valeur de **ALUOp**
- Sort soit un résultat de test sur 1 bit, soit un résultat d'opération sur 64 bits



4-b Instructions arithmétiques

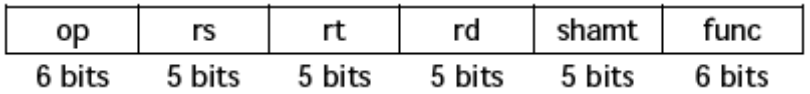
Banc des registres

- prend deux numéros de registres lus et retourne leur valeur
- prend un numéro de registre écrit et une valeur (pour la ranger dans le registre). Ce registre pour la plupart des instructions est le même qu'un des registres lus
- prend un signal **RegWrite:** 0 pas de registre écrit, 1 sinon



4-b Instructions arithmétiques

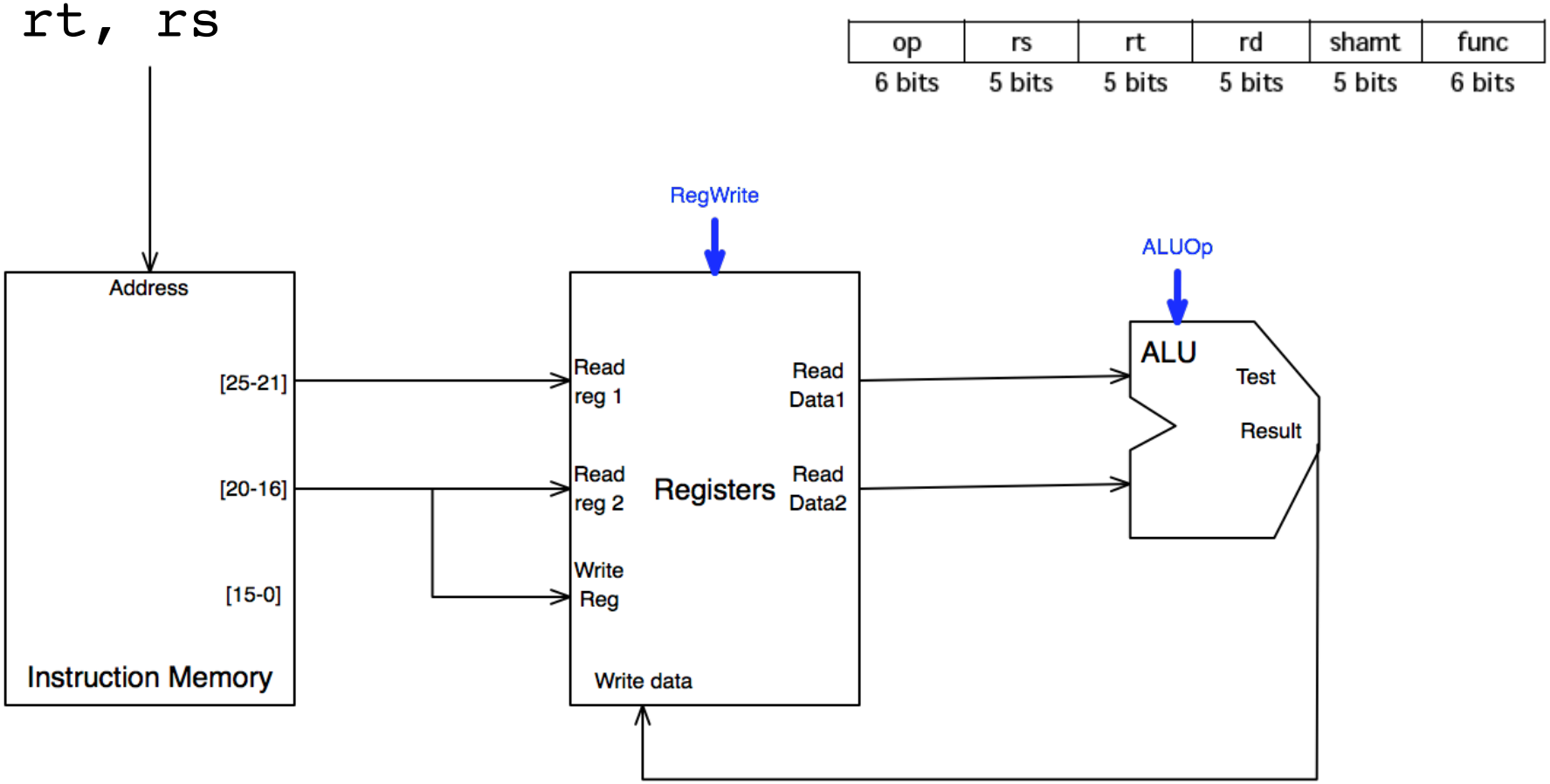
Etapes de l'instruction `OP rs, rt`



- OP est **`add, sub, or, and`**
- Extraire les numéros des registres rt et rs de l'instruction et les mettre sur les ports ReadRegister du banc de registre
- Extraire le numéro du registre rs, envoyer vers le port WriteRegister (avec RegWrite à 1)
- Envoyer les deux sorties ReadValue du banc de registre vers l'ALU
- Mettre la sortie de l'ALU vers le port WriteValue du banc de registre.

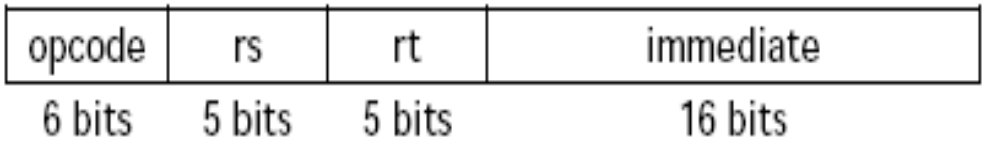
4-b Instructions arithmétiques

```
add rt, rs
```



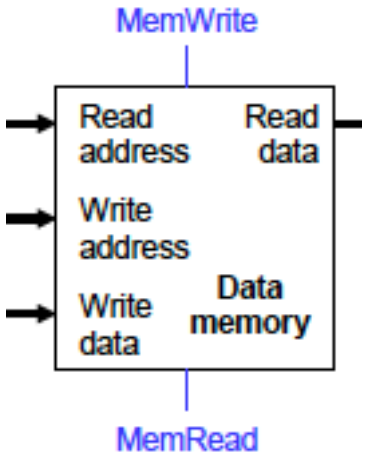
4-c Instructions d'accès mémoire

Mémoire de données



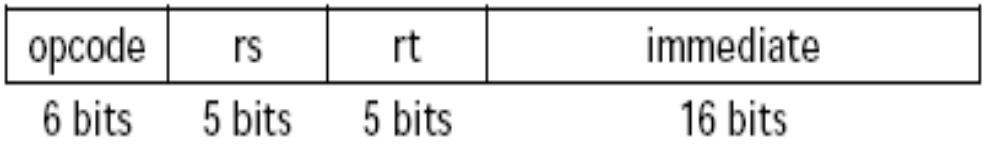

```
mov [rs+ imm], rt

mov rt, [rs+ imm]
```

- Accès en lecture:
  - Readaddress contient l'adresse à lire
  - **MemRead** est à 1. La valeur Readdata est le contenu de la cellule mémoire
- Accès en écriture:
  - Writeaddress contient l'adresse à écrire, Writedata la valeur à écrire
  - **MemWrite** à 1



4-c Instructions d'accès mémoire

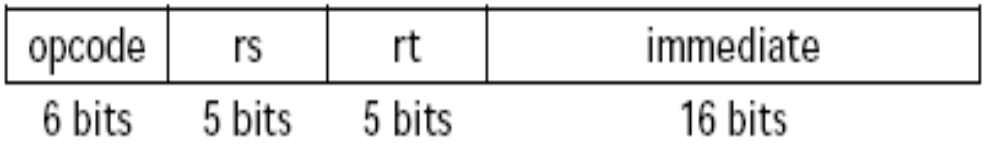
Etapes d'une instruction mv



- Extraire l'adresse et le numéro du registre rs de l'instruction
- Extraire le numéro de rt et le mettre dans le port WriteRegister du banch de registres
- Faire la somme avec l'ALU de l'adresse et de la valeur de rs pour avoir l'adresse de lecture
- Mettre la valeur de rt vers le port WriteValue du banc de registre.
- `mv registre, (registre + address)`
- Cette instruction s'appellera mv (load)

4-c Instructions d'accès mémoire

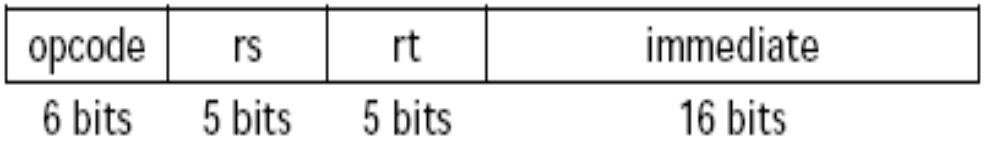
Etapes d'une instruction mv



- Extraire l'adresse et le numéro du registre rs de l'instruction
- Faire la somme avec l'ALU de l'adresse et de la valeur de rs pour avoir l'adresse de lecture
- Mettre la valeur de rt, en sortie du banc de registres vers le port WriteData de la mémoire.
- `mv (registre + address), registre`
- Cette instruction sera mv (store)

4-c Instructions d'accès mémoire

Chemin d'exécution

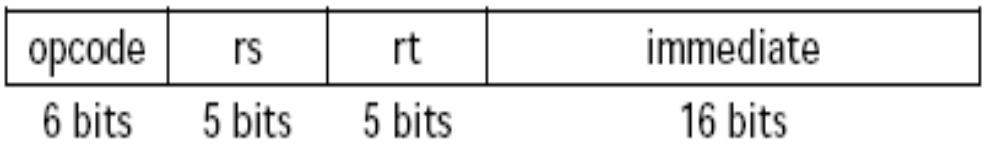


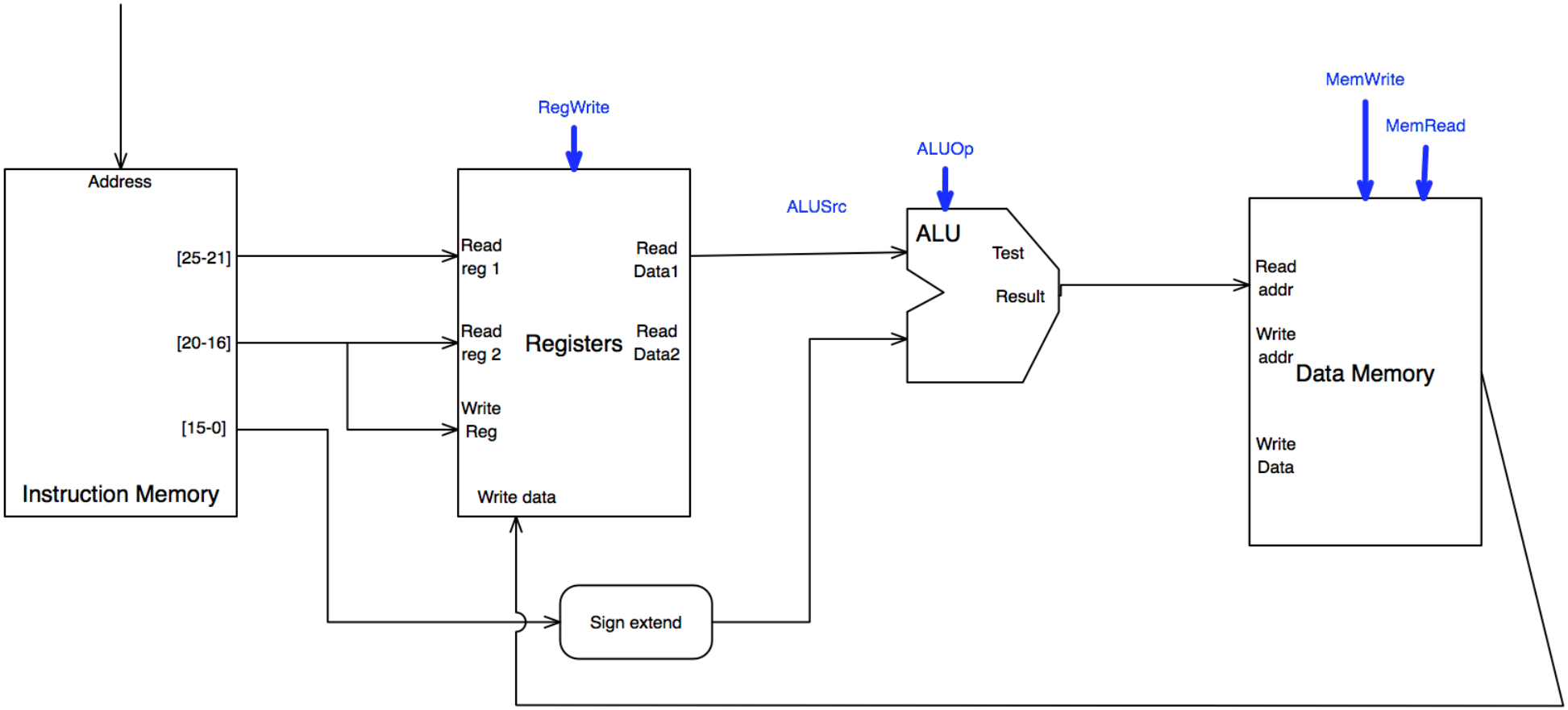
Multiplexeur: sélectionne une entrée parmi plusieurs à propager en sortie, suivant la valeur d'un signal de contrôle (définition limitée, au cas chemin d'exécution)

Signe étendu de 16 à 64 bits: on complète le nombre avec des 1 s'il est négatif, avec des 0 s'il est positif.

4-c Instructions de lecture mémoire

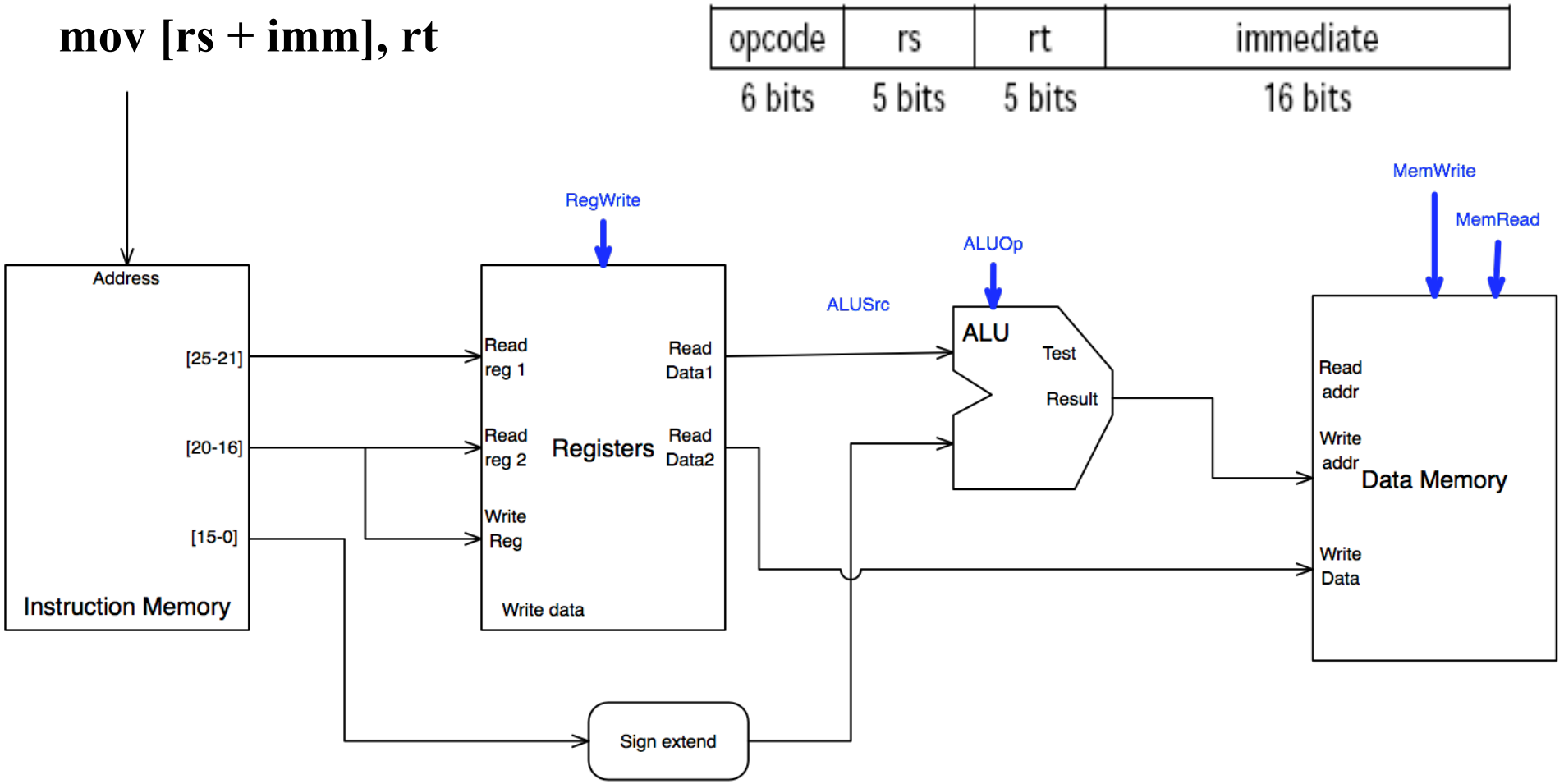
mov rt, [rs + imm]





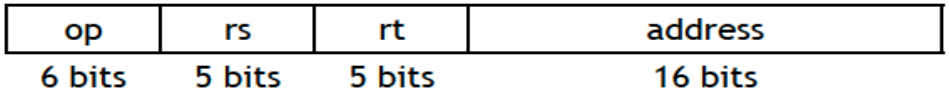
4-c Instructions d'écriture mémoire

mov [rs + imm], rt



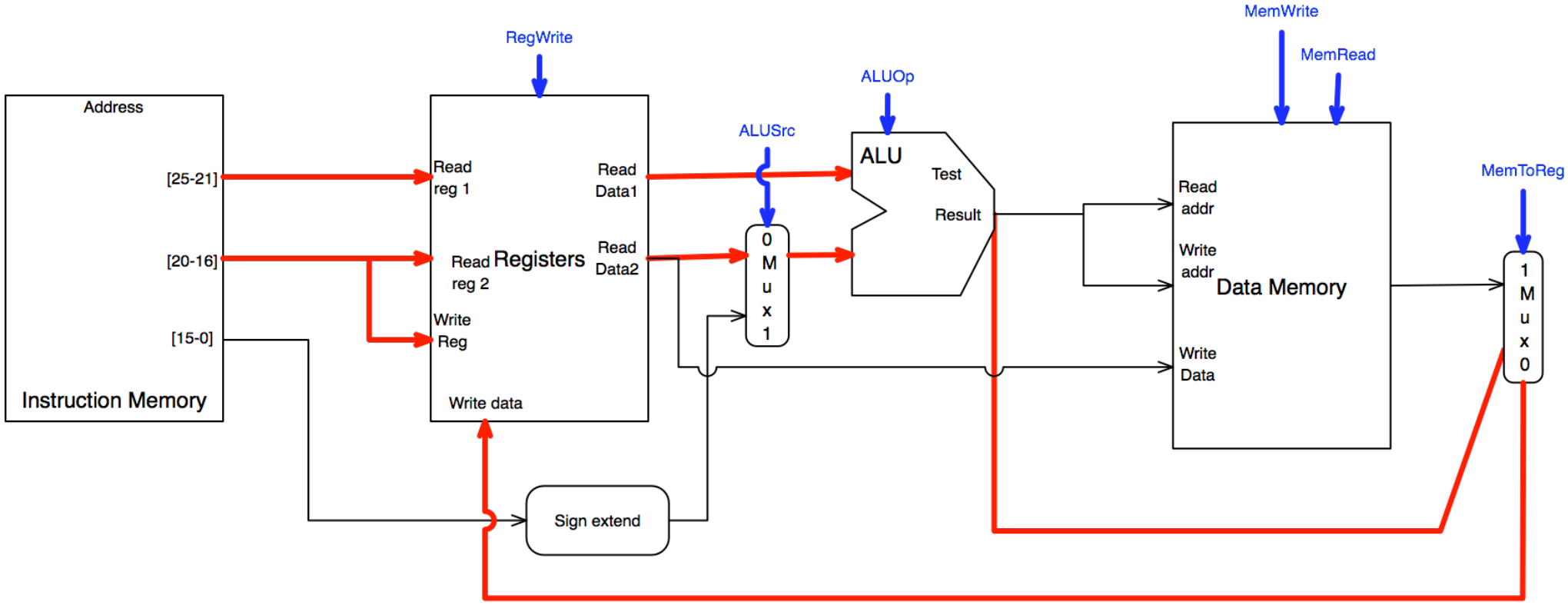
4-c Instructions d'accès mémoire

Chemin d'exécution

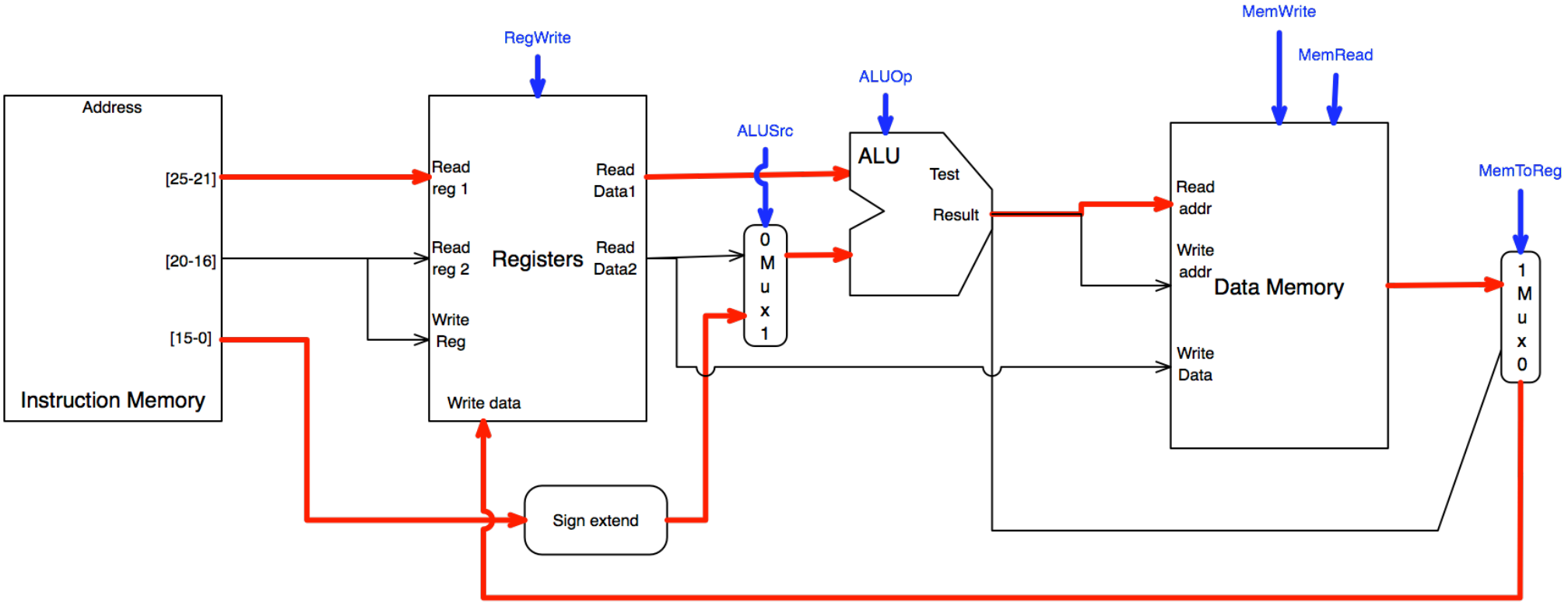


- l'ALU sort une valeur qui est soit
  - Stockée dans un registre (cas instructions arithmétiques)
  - Dirigée vers un port d'adresse de la mémoire, ReadAddress pour l'instruction de lecture, WriteAddress pour l'instruction d'écriture.

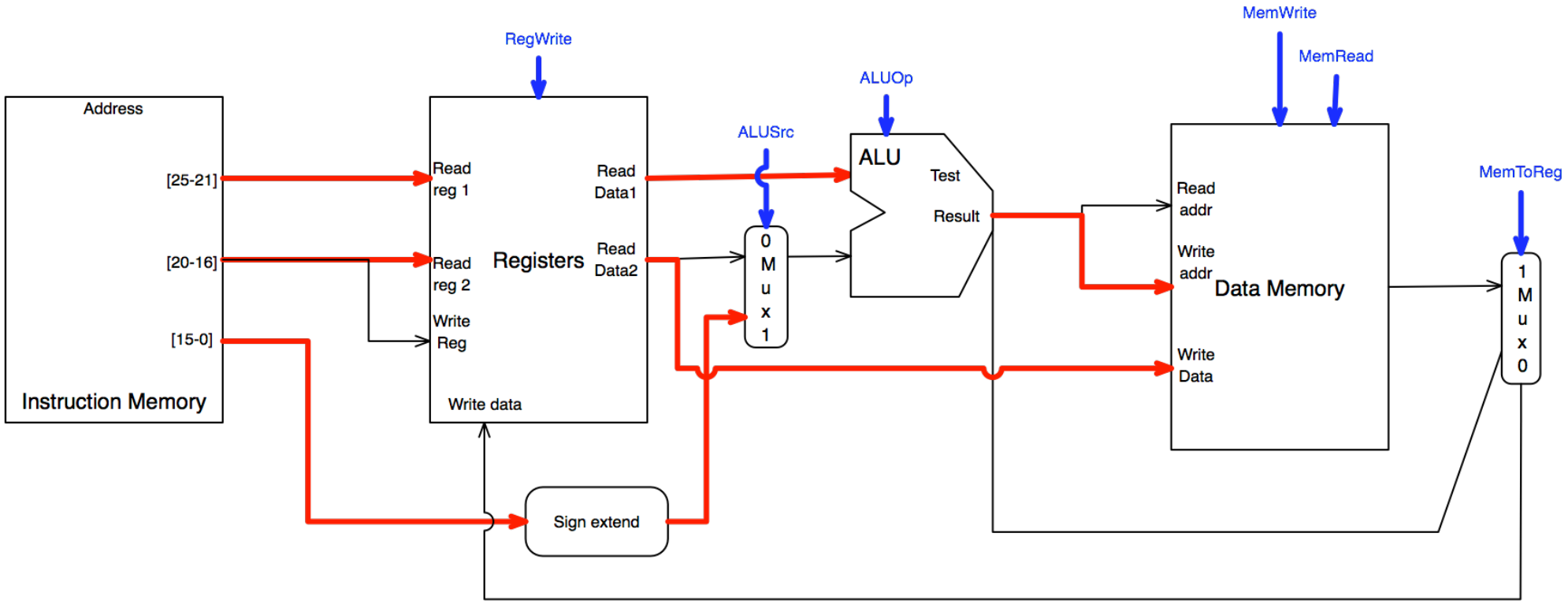
4-c Instructions arithmétiques et mémoire: ensemble sur le même circuit



4-c Instructions arithmétiques et mémoire: ensemble sur le même circuit

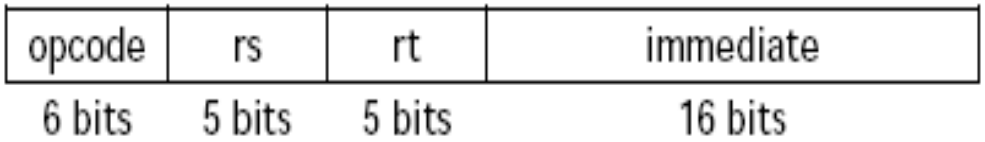


4-c Instructions arithmétiques et mémoire: ensemble sur le même circuit



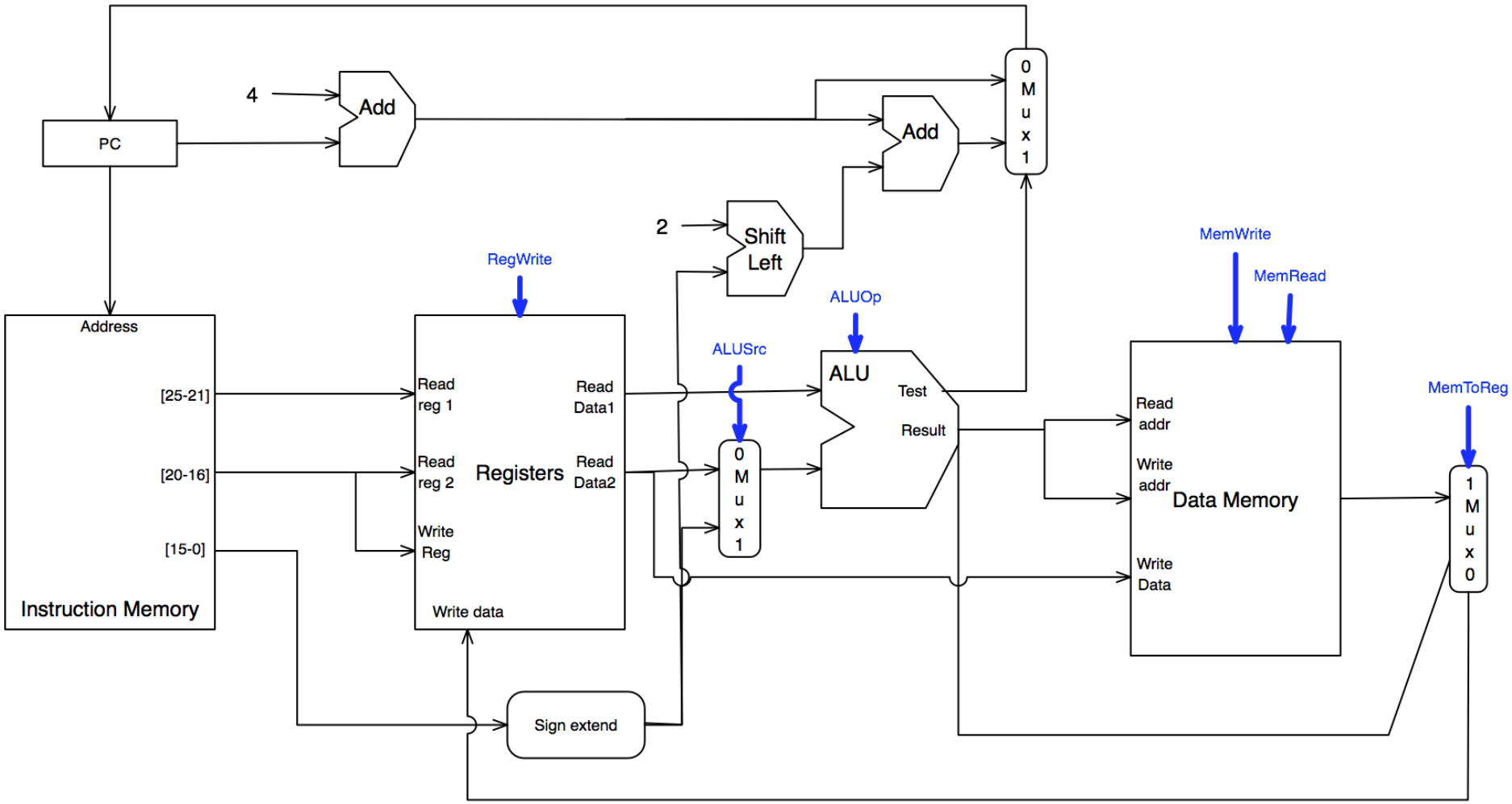
4-d Instructions de branchement

Etapes d'un jeq ou jne



- Tester le registre d'état avec l'ALU (ALUOp)
- Si comparaison du branchement vérifiée, ajouter le déplacement extrait de l'instruction à l'adresse
  - Le PC doit être mis à PC+4+(4*immédiat).
- Sinon, PC = PC+4. Test choisit entre les deux solutions en fonction résultat ALU, en fonction de ALUOp

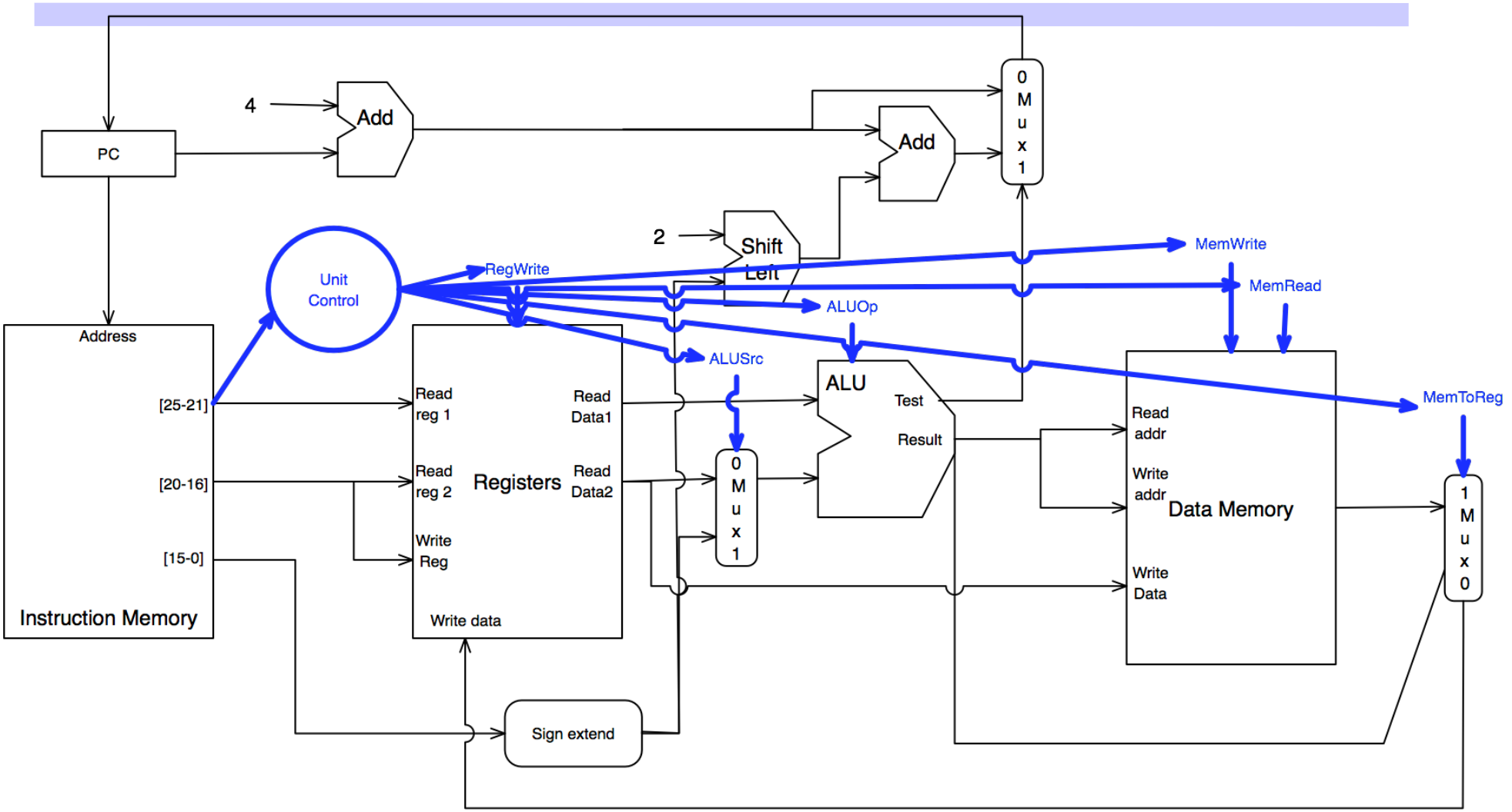
4-d Instructions de branchement



4-e Unité de contrôle

- Calcule toutes les signaux controllant les différentes unités
  - RegWrite, AluOp, AluSrc, MemRead, MemWrite, MemToReg
  - Dépend de l'instruction elle-même ou d'un résultat de l'ALU

4-e Unité de contrôle



4- Conclusion

A retenir:

- La microarchitecture déterminée par jeu d'instruction de l'ISA
- Certains choix d'ISA simplifient la microarchitecture
  - Instructions de même longueur
  - Numéros de registres au même endroit
- Banc de registres, unité mémoire, ALU et chemin d'exécution pour instruction sont communs à beaucoup d'architectures