TD 3 4MMARCA, Ensimag 2A/S6, ISI/SEOC

# Évaluation quantitative et processeur mono-cycle

Architecture avancée, 2018-2019

*Everyone knows Amdahl's Law but quickly forgets it !*
Thomas Puzak, IBM Research

## Ex. 1 : Loi d'Amdahl et CPI

On utilise un benchmark prenant 10 secondes, sachant que la moitié de ce temps est passé à faire du calcul flottant.
**Question 1 –** Calculez le temps d'exécution du benchmark sur une machine ayant une unité flottante (FPU) 5 fois plus rapide, et l'accélération résultante.

Une machine peut avoir soit une multiplication 4 fois plus rapide, soit des accès mémoire 2 fois plus rapides. Un benchmark composé de 20% de multiplications, de 50% d'accès mémoire, et de 30% du reste, prend 100 s.
**Question 2 –**

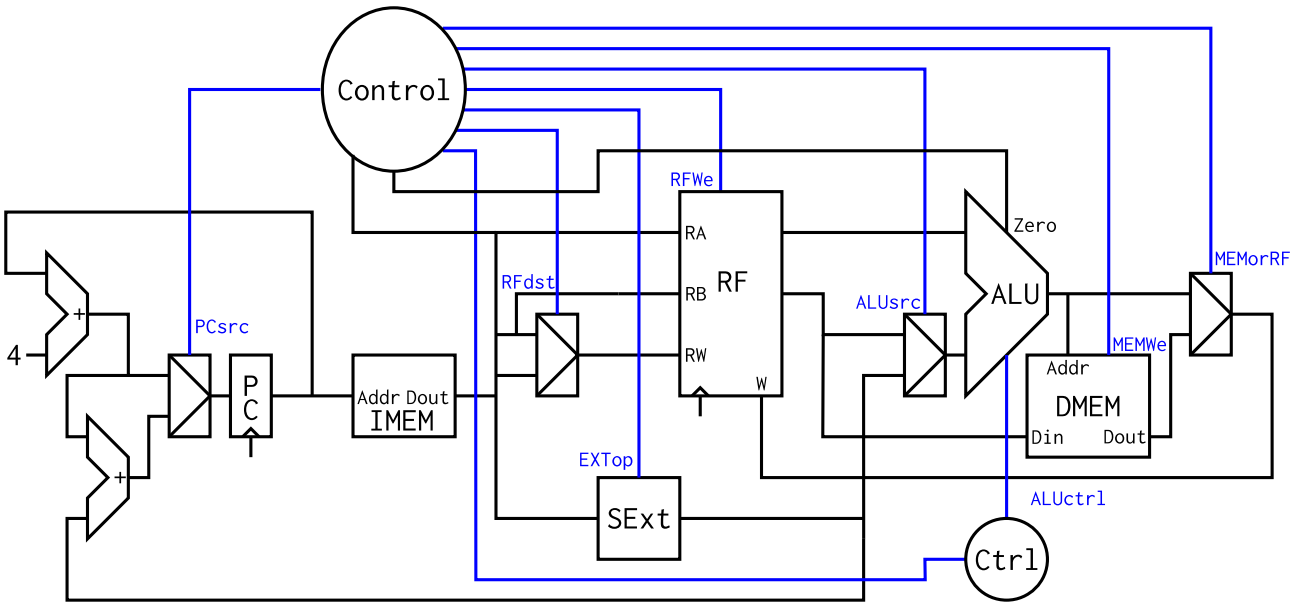
1. Donnez le temps d'exécution si l'on choisit d'accélérer la multiplication ;
2. Donnez le temps d'exécution si l'on choisit d'accélérer les accès mémoire ;
3. Donnez le temps d'exécution si l'on choisit d'accélérer les deux.

Soit une machine avec : 25% d'opérations flottantes, de CPI 4 ; 2% de racines carrées, de CPI 20 ; reste des opérations de CPI 1,33.
**Question 3 –** Calculez le CPI de la machine.

## Ex. 2 : Processeur mono-cycle

On suppose un processeur 3-adresses avec le jeu d'instruction minimal suivant : `addu $rd, $rs, $rt`, `subu $rd, $rs, $rt`, `lw $rt, sx_imm16($rs)`, `sw $rt, sx_imm16($rs)`, `bne $rt, $rs, pc + sx_imm16 * 4`, `ori $rt, $rs, zx_imm16`.

Soit le processeur mono-cycle suivant :




**Question 1 –** Complétez ce schéma en mettant sur chaque fil son « nom » et l'intervalle de bits qu'il utilise. On utilisera pour ce faire par exemple la notation `nom[7:2]`.

**Question 2 –** Donnez, pour chacune des instructions ci-dessus, la valeur des différents signaux de contrôle permettant d'en réaliser le comportement.

On s'intéresse tout d'abord à la boîte SEXop.

**Question 3 –** Listez les différentes opérations que cette boîte réalise. Proposez un codage du signal `EXtop` pour les réaliser, en prenant en compte l'encodage des instructions fournies en annexe. On suppose à présent que l'on veut réaliser *toutes* les instructions du premier tableau qui utilisent un immédiat.

**Question 4 –** Pour les différentes extensions :

- énumérez les conditions permettant de réaliser les différentes extensions ;
- il y a un bit qui est constant pour toutes les conditions, à quoi peut-il bien servir ?
- donnez les équations (simplifiées) permettant de calculer le type d'extension à faire.

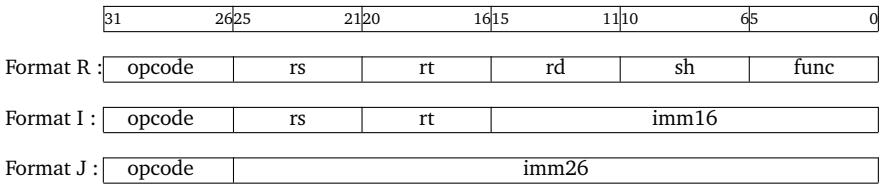
On va à présent ajouter le matériel nécessaire pour supporter quelques nouvelles instructions. Précisez également quels nouveaux fils de contrôle sont nécessaires, et comment les calculer, s'il y en a.

**Question 5 –** Ajouter le matériel nécessaire pour :

- support de l'instruction `j imm`, *jump*, de format J, qui fait :
  `pc = concat(pc[31:24], imm26 * 4);`
- support de l'instruction `jal imm`, *jump and link*, de format J, qui fait :
  `$31 = pc` et `pc = concat(pc[31:24], imm26 * 4);`
- support de l'instruction `jm imm($rs)`, *jump memory*, de format I, qui fait :
  `pc = MEMW[$rs+sx_imm16*4];`
- support de l'instruction `jalm rt, imm($rs)`, *jump and link memory*, de format I, qui fait :
  `$rt = pc` et `pc = MEMW[$rs+sx_imm16*4];`
- et pour finir, support de l'instruction `lrw $rd, $rs, $rt`, qui fait :
  `$rd = MEMW[$rs + $rt].`

# Formats et codes des instructions

Une instruction MIPS est un mot de 32 bits. Il y a trois catégories (*formats*) d'instructions :




Les tables suivantes présentent sous forme compacte le codage des différentes instructions.

Champ **opcode**

| 31...29 \ 28...26 | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 000 | **special** | **regimm** | j | jal | beq | bne | blez | bgtz |
| 001 | addi | addiu | slti | sltiu | andi | ori | xori | lui |
| 010 | **cop0** | - | - | - | - | - | - | - |
| 011 | - | - | - | - | - | - | - | - |
| 100 | lb | lh | - | lw | lbu | lhu | - | - |
| 101 | sb | sh | - | sw | - | - | - | - |
| 110 | - | - | - | - | - | - | - | - |
| 111 | - | - | - | - | - | - | - | - |

Champ **func**, lorsque l' **opcode** vaut **special**.

| 5...3 \ 2...0 | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 000 | sll | - | srl | sra | sllv | - | srlv | srav |
| 001 | jr | jalr | - | - | syscall | break | - | - |
| 010 | mfhi | mthi | mflo | mtlo | - | - | - | - |
| 011 | mult | multu | div | divu | - | - | - | - |
| 100 | add | addu | sub | subu | and | or | xor | nor |
| 101 | - | - | slt | sltu | - | - | - | - |
| 110 | - | - | - | - | - | - | - | - |
| 111 | - | - | - | - | - | - | - | - |

Champ **rt**, lorsque l' **opcode** vaut **regimm**.

| 20...19 \ 18...16 | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 00 | bltz | bgez | - | - | - | - | - | - |
| 01 | - | - | - | - | - | - | - | - |
| 10 | bltzal | bgezal | - | - | - | - | - | - |
| 11 | - | - | - | - | - | - | - | - |

Champ **rs**, lorsque l' **opcode** vaut **cop0**.

| 25...24 \ 23...21 | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 00 | mfc0 | - | - | - | mtc0 | - | - | - |
| 01 | - | - | - | - | - | - | - | - |
| 10 | - | - | - | - | - | - | - | - |
| 11 | - | - | - | - | - | - | - | - |

Champ **func**, lorsque l' **opcode** vaut **cop0**.

| 5...3 \ 2...0 | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 000 | - | tlbr | tlbwi | - | - | - | tlbwr | - |
| 001 | tlbp | - | - | - | - | - | - | - |
| 010 | rfe | - | - | - | - | - | - | - |
| 011 | - | - | - | - | - | - | - | - |
| 100 | - | - | - | - | - | - | - | - |
| 101 | - | - | - | - | - | - | - | - |
| 110 | - | - | - | - | - | - | - | - |
| 111 | - | - | - | - | - | - | - | - |