



**POLYTECHNIQUE  
MONTREAL**

# INF1600

## Architecture des micro-ordinateurs

Laboratoire 5

Soumis par:  
Ioana Daria Danciu - 2081744  
Alexandre Gélinas - 2083465

Groupe de Laboratoire:  
02

Le 25 avril 2020

## 1.2 Barème

TP5		/4,00
Q1	/1,00	
Q2	/1,00	
Q3	/1,00	
Q4	/1,00	
Q5 (bonus)	/1,00	

**Q1:**

INSTRUCTION/TYPE D'INSTRUCTION	CPI
OP_ALU	3
LDI	3
READ_MEM	4
WRITE_MEM	4

**Q2:**

Nous avons changé l'instruction suivant le decode pour fetch pour réduire le CPI à 2 (L3)

```

when decode =>
  case inst is
    when inst_alu      => state <= fetch;
    when inst_read_mem => state <= read_mem;
    when inst_write_mem => state <= write_mem;
    when inst_loadi    => state <= ldi;
    when inst_branch   => state <= jump;
    when inst_stop     => state <= stop;
    when others        => state <= stop;
  end case;

```

Nous avons changé le process pour pouvoir faire une réaction lorsque notre s\_op\_ual change. De plus, nous avons modifié la condition pour faire les instructions de OP\_ALU pour qu'il teste si la valeur du signal a changé pour '1'.

```

process( state, s_op_ual, rmem_confirmed ) is
begin
  wFLAG <= '0';
  if( s_op_ual = '1' ) then
    choixSource <= 0;
    wreg        <= '1';
    wFLAG       <= '1';
  elsif( rmem_confirmed = '1' ) then
    choixSource <= 1;
    wreg        <= '1';
  elsif( state = ldi ) then
    choixSource <= 2;
    wreg        <= '1';
  else
    choixSource <= 3;
    wreg        <= '0';
  end if;
end process;

```

**Q3:**

Afin de réduire le CPI de `read_mem` de 1 cycle d'horloge, nous avons retiré la condition ci-dessous, qui faisait en sorte que le programme passait à l'instruction `fetch` que lorsque l'instruction `read_mem` a été complétée. Ainsi, lorsqu'on est en train de lire dans la mémoire, on va également chercher la prochaine instruction, ce qui permet d'éviter de passer par un registre intermédiaire et de réduire le CPI de 1.

```
when read_mem =>
  --if( rmem_confirmed = '1' ) then
    state <= fetch;
  --end if;
```

**Q4:**

Afin de réduire le CPI de `write_mem` à deux cycles d'horloge, nous avons remplacé le changement d'état de `write_mem` à `fetch`. Cela permet d'aller directement chercher la prochaine instruction au moment où l'on comment à écrire dans la mémoire. Étant donné qu'il n'y aucune autre instruction après le `write_mem`, le programme n'est pas affecté et nous obtenons la même valeur.

```
when decode =>
  case inst is
    when inst_alu      => state <= fetch;
    when inst_read_mem => state <= read_mem;
    when inst_write_mem => state <= fetch;
    when inst_loadi    => state <= ldi;
    when inst_branch   => state <= jump;
    when inst_stop     => state <= stop;
    when others        => state <= stop;
  end case;
when read_mem =>
```

**Q5 (bonus):**

Il est impossible de pouvoir faire un CPI de 2 pour le `read_mem` étant donné que l'opération de mettre A à MA prend un cycle autre que ceux de base. Ainsi, nous n'obtenons pas notre donnée assez rapidement pour pouvoir l'utiliser sans encombrer les prochaines instructions.