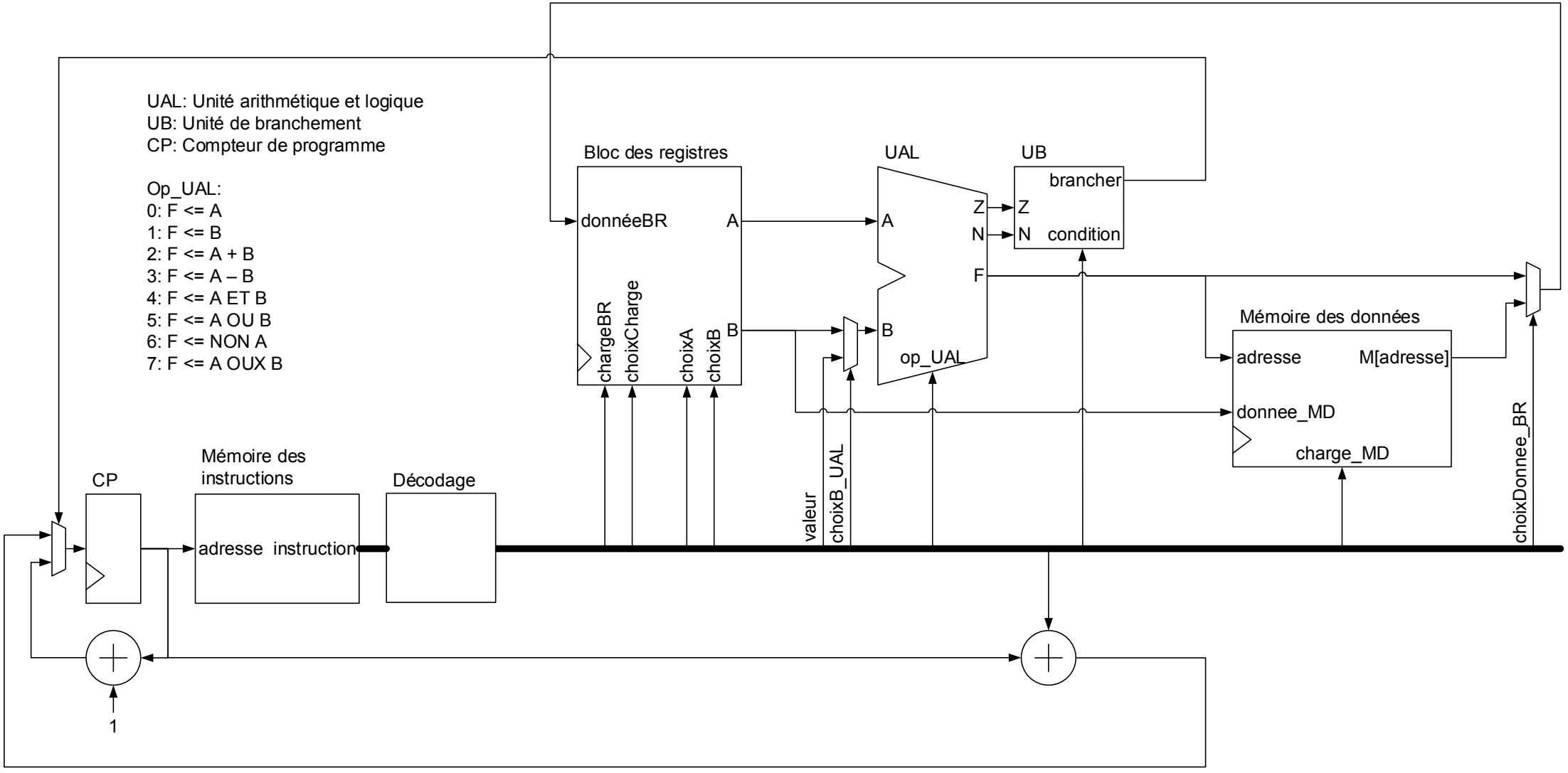
Modélisation VHDL de l'unité de contrôle d'un processeur à usage général

Modélisation VHDL de l’unité de contrôle d’un processeur à usage général

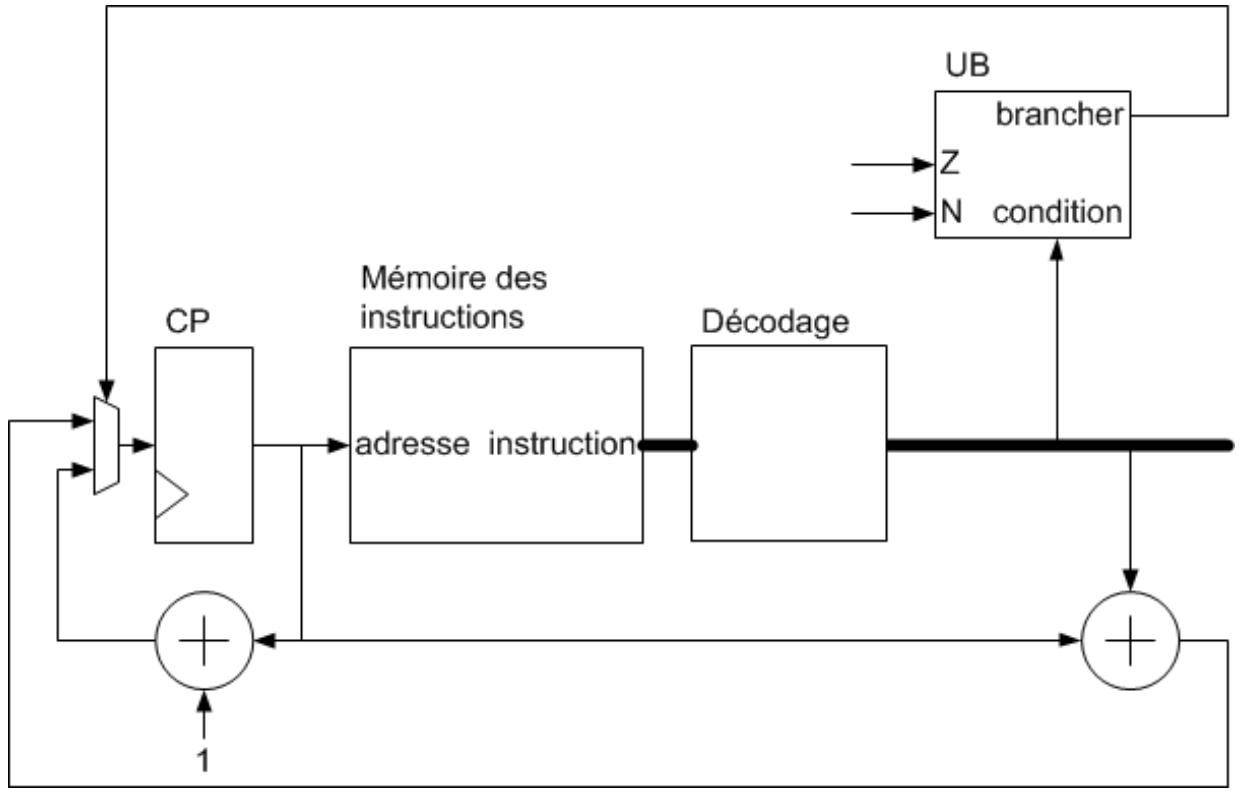
- Rappel: architecture du processeur PolyRISC et son chemin des données
- Modélisation du compteur de programme
- Modélisation de la mémoire des instructions
- Modélisation de l’unité de branchement
- Modélisation du décodage

Le processeur à usage général PolyRISC



Unité de contrôle du processeur à usage général

- L'unité de contrôle dirige le chemin des données.
- Les composantes de l'unité de contrôle sont :
  - un compteur de programme (*Program Counter* – PC) qui pointe à la prochaine instruction à exécuter;
  - une mémoire des instructions contenant le programme à exécuter;
  - un module pour décoder l'instruction à exécuter, c'est-à-dire de dériver les signaux de contrôle du chemin des données à partir de l'instruction;
  - un module pour déterminer si un branchement a lieu ou non;
  - deux additionneurs pour calculer l'adresse de la prochaine instruction.



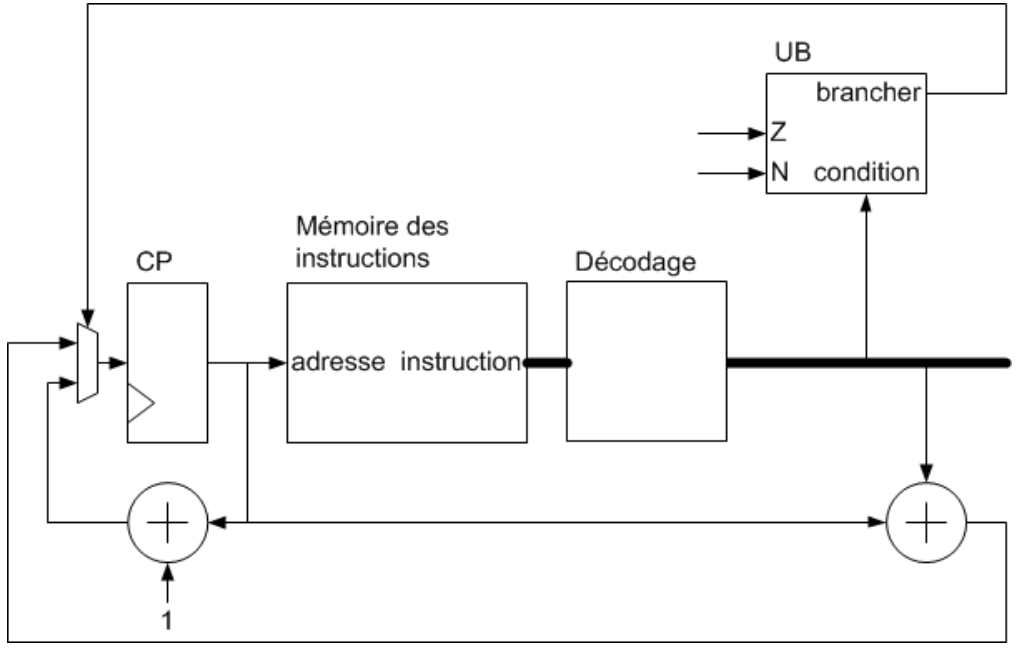
Encodage des instructions en mémoire

- Considérations pour le format des instructions et de leur encodage en mémoire
  - C'est un problème à plusieurs solutions possibles.
  - On favorise la régularité, ce qui simplifie le décodage et accélère le processeur.
  - L'encodage, le jeux d'instructions et les modes d'adressage sont intimement liés.
- Choix effectués ici:
  - Encodage sur un seul mot de 32 bits: chaque instruction s'exécute en un seul cycle.
  - Le mot d'instruction est divisé en 6 champs.
  - Modes d'adressage limités conformément à la philosophie RISC.

Instruction	bits 31-30 catégorie	bits 29:26 détails	bits 25:21	bits 20:16	bits 15:5	bits 4:0
RC := RA ◊ RB	00	code de ◊	RC	RA	-	RB
RC := RA ◊ valeur	01	code de ◊	RC	RA	valeur(15:0)	
si (RA ◊ RB) goto (CP + valeur)	10	code de ◊	RB	RA	valeur(15:0)	
RC := MD[RA + valeur]	11	0000	RC	RA	valeur(15:0)	
MD[RA + valeur] := RB	11	0001	RB	RA	valeur(15:0)	

Compteur de programme

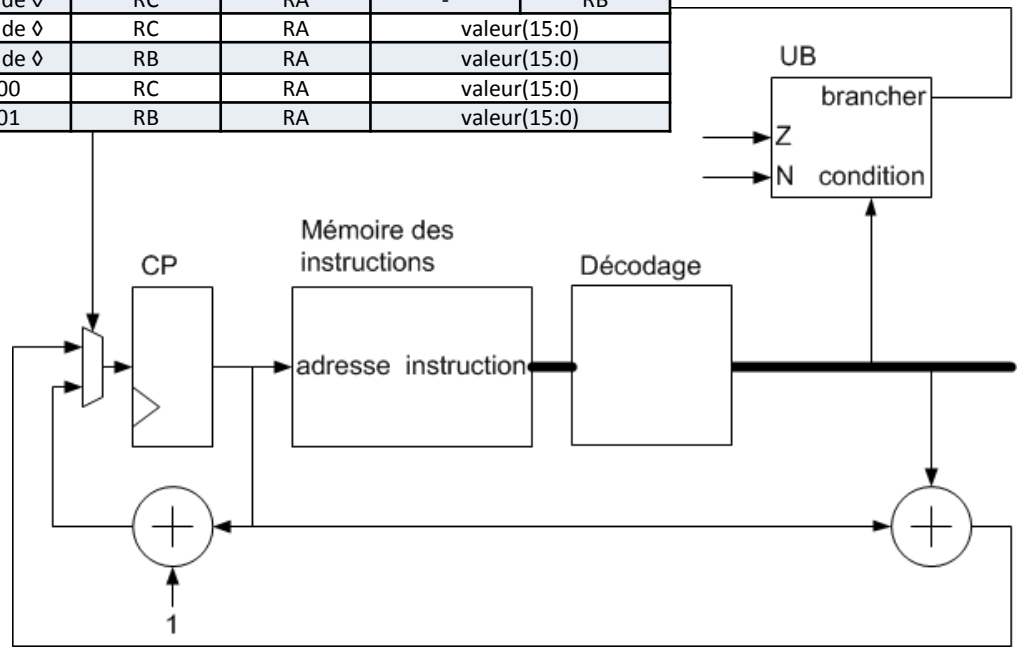
```
-- dans la partie déclarative de l'architecture
signal CP : integer range 0 to (2 ** Mi - 1); -- le compteur de programme


-- dans le corps de l'architecture
  process (CLK, reset)
  begin
    if rising_edge(CLK) then
      if reset = '1' then
        CP <= 0;
      else
        if brancher = '1' then
          CP <= CP + instruction.valeur;
        else
          CP <= CP + 1;
        end if;
      end if;
    end if;
  end process;
```



Mémoire des instructions

Instruction	bits 31-30 catégorie	bits 29:26 détails	bits 25:21	bits 20:16	bits 15:5	bits 4:0
RC := RA ◊ RB	00	code de ◊	RC	RA	-	RB
RC := RA ◊ valeur	01	code de ◊	RC	RA	valeur(15:0)	
si (RA ◊ RB) goto (CP + valeur)	10	code de ◊	RB	RA	valeur(15:0)	
RC := MD[RA + valeur]	11	0000	RC	RA	valeur(15:0)	
MD[RA + valeur] := RB	11	0001	RB	RA	valeur(15:0)	

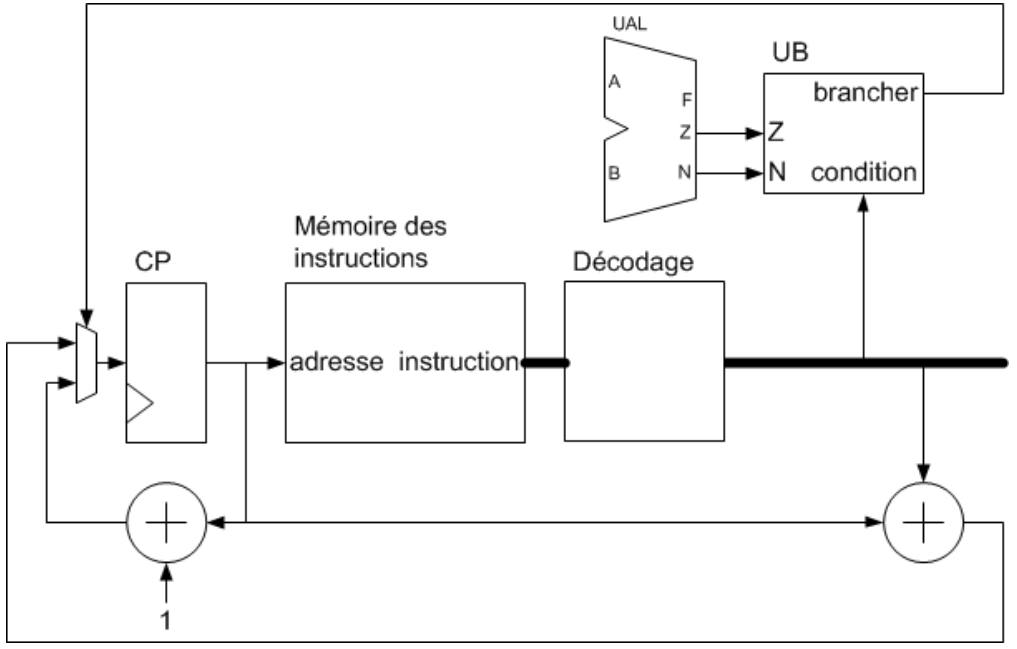
```
-- structure pour l'encodage d'une instruction
type instruction_type is record
   categorie : natural range 0 to 3;
   details : natural range 0 to 15;
   reg1 : natural range 0 to Nreg - 1;
   reg2 : natural range 0 to Nreg - 1;
   valeur : integer range -2 ** (WImm - 1) to 2 ** (WImm - 1) - 1;
end record;
signal instruction : instruction_type;

-- instructions prédéfinies
constant NOP : instruction_type := (branchement, jamais, 0, 0, 0);
constant STOP : instruction_type := (branchement, toujours, 0, 0, 0);

-- mémoire des instructions et définition du programme
type memoireInstructions_type is array (0 to 2 ** Mi - 1) of instruction_type;
constant memoireInstructions : memoireInstructions_type := (
(reg_valeur, passeB, 2, 0 , 0),  --  R2 := 0;
(reg_valeur, passeB, 0, 0, 12),  --  R0 := 12;
(reg_valeur, passeB, 1, 0 , 7),  --  R1 := 7;
(reg, AplusB, 0, 0 , 1),         --  R0 := R0 + R1;
(reg, AouxB, 1, 0 , 1),          --  R1 := R0 OUX R1;
(reg_valeur, AplusB, 1, 1, -3),  --  R1 := R1 - 3;
(branchement, pgq, 2, 1, -1),    --  si R1 > 0 (R2) goto -1;
STOP,
others => NOP
);
```



Une instruction est définie comme une structure à 5 champs. La mémoire des instructions est définie dans la partie déclarative de l'architecture comme un tableau constant dans lequel on place les valeurs des instructions du programme.

Unité de branchement

```
-- dans la partie déclarative de l'architecture
signal condition : natural range 0 to 7;
constant egal : natural := 0; constant diff : natural := 1;
constant ppq : natural := 2; constant pgq : natural := 3;
constant ppe : natural := 4; constant pge : natural := 5;
constant toujours : natural := 6; constant jamais : natural := 7;

signal brancher : std_logic;
signal condition : conditionBranchement_type;


-- dans le corps de l'architecture
  process(Z, N, condition)
  begin
    case condition is
      when egal => brancher <= Z;
      when diff => brancher <= not(Z);
      when ppq => brancher <= N;
      when pgq => brancher <= not(N) and not(Z);
      when ppe => brancher <= N or Z;
      when pge => brancher <= not(N) or Z;
      when toujours => brancher <= '1';
      when jamais => brancher <= '0';
    end case;
  end process;
```



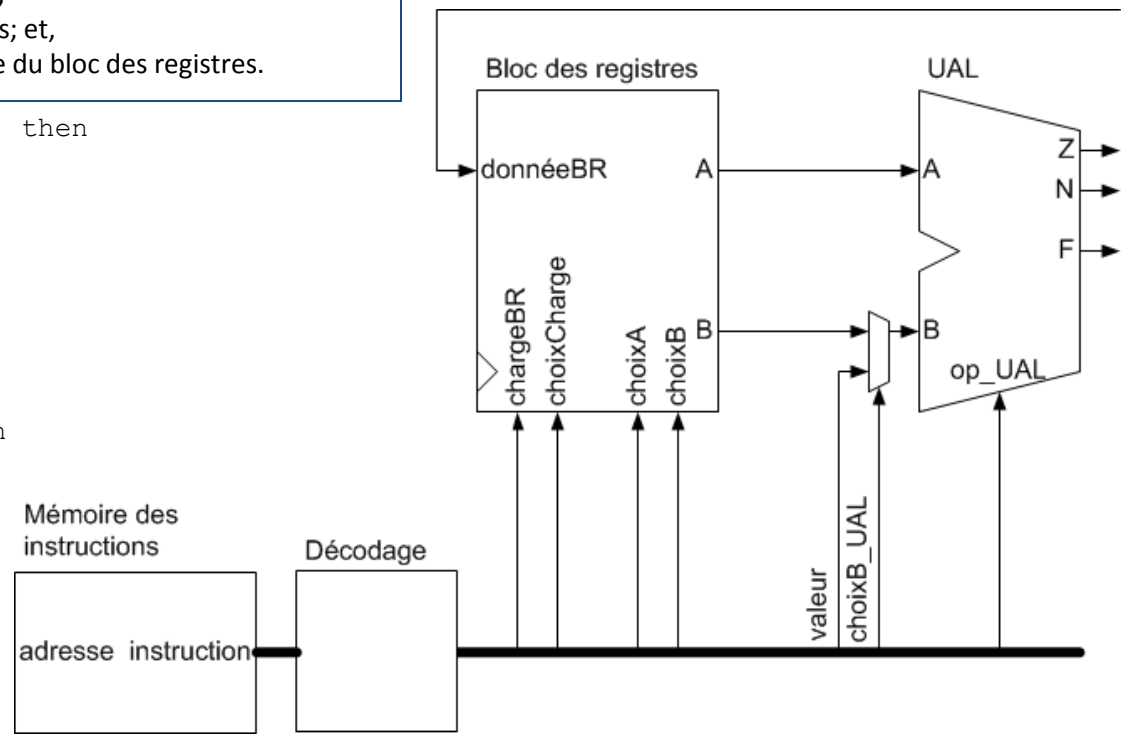

```
process (instruction)
begin
   if (instruction.categorie = branchement) then
      condition <= instruction.details;
   else
      condition <= jamais;
   end if;
end process;
```

Décodage

```
instruction <= memoireInstructions(CP);

process (instruction)
begin

-- pour le bloc des registres
if (instruction.categorie = reg or
   instruction.categorie = reg_valeur or
   (instruction.categorie = memoire and instruction.details = lirememoire)) then
   chargeBR <= '1';
else
   chargeBR <= '0';
end if;
choixCharge <= instruction.reg1;
-- etc. ...

-- pour l'UAL
if (instruction.categorie = reg or instruction.categorie = reg_valeur) then
   op_UAL <= instruction.details;
elsif (instruction.categorie = branchement) then
   -- pour faire la comparaison entre les opérandes
   op_UAL <= AmoinsB;
else
   -- lire et écrire la mémoire, calcul de l'adresse effective
   op_UAL <= AplusB;
end if;
-- etc. ...
```

Le décodage des instructions produit les signaux de contrôle:

- du bloc des registres;
- de l'UAL;
- de l'unité de branchement;
- de la mémoire des données; et,
- du multiplexeur de l'entrée du bloc des registres.



Vous devriez maintenant être capable de …

- Donner un diagramme montrant les composantes et leurs connexions de l'unité de contrôle d'un processeur à usage général. (B2)
- Donner le code VHDL pour le compteur de programme, la mémoire des instructions et l'unité de branchement d'un processeur à usage général. (B3)

Code	Niveau (http://fr.wikipedia.org/wiki/Taxonomie_de_Bloom)
B1	Connaissance – mémoriser de l'information.
B2	Compréhension – interpréter l'information.
B3	Application – confronter les connaissances à des cas pratiques simples.
B4	Analyse – décomposer un problème, cas pratiques plus complexes.
B5	Synthèse – expression personnelle, cas pratiques plus complexes.