Juin 2013 AUCUN DOCUMENT M1 AE

Exercice 1 : 4 pts

On dispose d'une mémoire principale de 2^{16} octets. Le transfert minimal entre la mémoire et le cache est de 8 octets. On utilise un cache direct mapping de 32 lignes.

1) Définissez le nombre de champs de l'adresse et la taille de chaque champ.
2) Dans quelle ligne se trouvent les mots dont les adresses sont :
   0001 0001 0001 1011
   1100 0011 0011 0100
   1101 0000 0001 1101
   1010 1010 1010 1010
3) Que range-t-on avec la donnée et pourquoi ?
4) Quelle est la taille effective du cache?

Exercice 2 (12pts)

On utilise une machine SIMD de 64 Processeurs Elémentaires (PE) à mémoire distribuée. Chaque processeur possède les registres A,B,C,D,I,R. Une unité de contrôle (ACU) pilote l'ensemble des PE. Et dispose de ses propres registres INX1, INX2, INX3, INX4, INX5. Les PE sont reliés par un réseau de communication.

On dispose des instructions suivantes

Instruction ACU :

MVI INXi ,#	move immédiat dans INXi
INC INXi	incrément de INXi : INXi <= INXi + 1
JLT INXi, INXj, Label	Branchement conditionnel si INXi<INXj goto Label
	LT less than GT greater than etc...
MASK Lit	masque les PE actifs par un littéral de 64 bits 1 actif 0 inactif
	Mask (0) pilote le PE0 Mask (1) pilote le PE1 etc…

Instruction SIMD :

VLOAD r	Adressage indirect indexé : r <= ((Di) + (Ii))
(3cycles)	r ∈ {Ai,Bi,Ci,Ri}
VMOV r1, r2	Transfert registres r1 <= (r2)
(1cycle)	r1,r2 ∈ {Ai,Bi,Ci,Di,Ii,Ri}
VMVI r,#	move immédiat r <= #
(2cycles)	r ∈ {Ai,Bi,Ci,Di,Ii,Ri}
VSTORE r	Adressage indirect indexé : (Di) + (Ii) <= (r)
(3cycles)	r ∈ {Ai,Bi,Ci,Ri}
VADD r1, r2	Addition registres r1 <= (r1) + (r2)
(2cycles)	r1,r2 ∈ {Ai,Bi,Ci,Di,Ii,Ri}
VADDI r,#	Addition immédiat r <= (r) + #
(3cycles)	r ∈ {Ai,Bi,Ci,Di,Ii,Ri}
LCYCLE r	left cyclic Ri-s <= (Ri) avec s = 2^d et d = (r)
(3cycles)	

Soit trois vecteurs A, B, C de 640 éléments rangés dans les mémoires locales de chaque PE.

1) Dessinez les 3 rangements de A, B et C dans les mémoires distribuées
   En bloc, en cyclique et par bloc de taille 2 (cycliquement des blocs de taille 2)

Donnez le code assembleur pour les codes suivants pour chacun des **trois** rangements, pour faciliter l'écriture des 9 codes les parties communes pourront être reprises une fois définie, par un carré englobant dans le codes suivants (en couleur)

2) For I = 0 to 639 step 2 A(I) = B(I) + C(I) I de 0 a 639 avec pas de 2
3) For I = 0 to 639 step 3 A(I) = B(I) + C(I)
4) For I = 0 to 63 S = S + A(I) dimension de A de 64 éléments S dans registre Ai

Exercice 3 (4 points)

Voici un code VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.numeric_std.all;

entity IP_Xxx is
GENERIC (Mycode : std_logic_vector(10 downto 0));
Port (
     Tin : in  STD_LOGIC_VECTOR (31 downto 0);
     IPcode : in  STD_LOGIC_VECTOR (10 downto 0);
     clk : in STD_LOGIC;
     reset: in STD_LOGIC;
     IPdone : out  STD_LOGIC);
end IP_Xxx;

architecture FSM of IP_xxx is
signal rst,  IPd : std_logic;
signal cptr_d, cptr_next : std_logic_vector (31
downto 0);
type state_type is (idle, starting, finish,
fetch_next, Decode_next);
signal next_state, current_statew: state_type;

COMPONENT reg1
 PORT(  load :  IN std_logic;
        d :     IN std_logic_vector(31 downto 0);
        clr ,clk: IN std_logic;
        q :  OUT std_logic_vector(31 downto 0));
 END COMPONENT;
begin
Xxx_reg : reg1
   port map (
      d(31 downto 0) => cptr_d,
      load => '1',
      clk => clk,
      clr => reset,
      q(31 downto 0) => cptr_next );

cptr_d <= Tin - 3 when rst='1' else cptr_next-1;
IPdone <= IPd;
```

```
state_reg: process(clk, reset)
begin
   if (clk'event and clk='1') then
      if (reset='1') then
         current_statew <= idle;
      else
         current_statew <= next_state;
      end if;
   end if;
end process;

comb_logic: process( Ipcode,cptr_next)
begin
   case current_statew is
      when idle =>
         if Ipcode =  Mycode then
            rst<='1';
            next_state <= starting;
         else
            rst <='0';
            next_state <= idle;
         end if;
         IPd <= '0';
      when starting =>
         rst<='0';
         if cptr_d= x"00000000" then
            next_state <= finish;
         else
            next_state <= starting;
         end if;
      when finish =>
         IPd  <= '1';
         next_state<= fetch_next;
      when fetch_next =>
         IPd  <= '0';
         next_state<= Decode_next;
      when Decode_next =>
         next_state<= idle;
      when others =>
         next_state <= idle;
   end case;
end process;
end FSM;
```

1)Décrire le comportement (un automate avec input et output) de la FSM représentée par les deux process state_reg et comb_logic

2)En supposant que Tin vaut x "0000000F" quelle sera la valeur rangée dans le registre Xxx_reg et à quel moment ?

3)A quel moment le signal IPdone est-il égal à '1' ? Que réalise cet IP ?

4)On propose d'attribuer la valeur "0000001111" à cet IP. Donnez les deux codes des instructions HoMade sur 16 bits.

- Avec la valeur de sommet de pile qui sert de Tin et qui est dépilée par l'instruction
- Idem mais sans dépiler la valeur sur la pile