**Problemă CN1**

**9 iunie**

Fie următorul cod în assembly (simplificat):

|  |
| --- |
| mov ax, $a  mov bx, $b  **\_while**: cmp ax, bx  jg **\_greater**  jl **\_less**  je **\_final**  jmp **\_while**  **\_greater**:  sub ax, bx *;ax = ax - bx*  jmp **\_while**  **\_less**:  sub bx, ax *;bx = bx - ax*  jmp **\_while**  **\_final**: *;se afiseaza conținutul lui ax* |

Avem 2 procesoare, fiecare având frecvența de **1Ghz,** care execută acest cod, procesoarele A și B. Metoda de *branch-prediction* folosită de fiecare este:

1. 1-bit counter (taken/not taken)
2. 2-bit counter (strongly taken/weak taken/weak not taken/strongly not taken)

Ambele pornesc cu o abordare pesimistă, counter-ul/state-machine-ul fiecăreia având starea inițială pentru procesorul A *not taken* și pentru B *strongly not taken*. De altfel, ambele implementează un *branch-predictor global*, astfel că există un singur counter/state-machine global, pentru toate branch-urile.

Se știu numărul de cicli de ceas pentru fiecare tip de instrucțiune: **mov** - 1 ciclu, **cmp** - 1 ciclu, **sub** - 1 ciclu, **branch** - 20 cicli (dacă nu a fost prezis corect) sau 2 cicli (dacă a fost prezis corect).

Se cer următoarele:

1. Pentru input-ul **a=26** și **b=32** să se evidențieze prin ce stări trec branch-predictor-ele celor 2 procesoare, semnalându-se și care branch-uri au fost prezise corect, respectiv incorect. **(4p)**
2. Să se calculeze timpul de rulare al programului pe cele 2 procesoare, pentru input-ul de la punctul 1. **(3p)**
3. FIe un procesor C, cu frecvență de 1Ghz, care are aceeași metoda de *branch-prediction* pe 2 biți ca procesorul B, implementând însă un *branch-predictor local* (câte un counter/state-machine pentru fiecare branch). Execută procesorul C codul dat mai repede decât procesorul A sau B, pentru input-ul de la punctul 1? Argumentați. Care este speed-up față de B? **(3p)**

**Problemă CN1**

**10 iunie**

Fie următorul cod assembly :

|  |  |  |
| --- | --- | --- |
| 0  1  2  3  4  5  6  7  8  9  10 | loop:  Dan: | load R0, #n  mov R1,0  mov R2, 1  cmp R0, 1  je Dan  mov R3, R1  add R1, R2  mov R2,R3  sub R0,1  j loop  ;print R1 |

Avem 2 procesoare, A și B:

* A = mașină, fără pipeline, cu branch prediction pe 1 bit.  
  Numărul de cicli /instrucțiuni este: load - 10 cicli, mov 1 ciclu, cmp 1 ciclu, je - 10 cicli la BP failed, je - 1 ciclu dacă a fost prezis corect , j - 1 ciclu , add/sub 4 cicli
* B = mașină cu pipeline în 5 stagii(IF,ID,E,M,W), 1 ciclu/stagiu  
  În cazul hazardelor de date, procesorul introduce ‘bubbles’.

Se presupune că [n] = 7.

Se cer următoarele:

1. **(0.5p)** Ce valoare se afișează?
2. **(3.5p)** Pentru inputul de mai sus să se evidențieze prin ce stări trece branch-predictorul procesorului A, semnalându-se și care branch-uri au fost prezise corect, respectiv incorect.   
   Câți cicli durează execuția programului pe procesorul A?
3. **(4.5p)** Evidențiați execuția instrucțiunilor în pipeline pentru prima iterație pe procesorul B(ex: cu ajutorul unei diagrame pipeline). Cum se modifică distribuția hazardelor după prima iterație?   
   Câți cicli durează execuția programului pe procesorul B?
4. **(1.5p)** Presupunând că procesorul A rulează la 500 MHz și procesorul B la 400Mhz câte microsecunde durează execuția pe fiecare procesor.

**Problemă CN1**

**11 iunie**

Fie următorul cod în assembly:

|  |
| --- |
| mov ax, $a  mov bx, $b  mov cx, $c  **loop**: cmp cx, 0  je **out**  add ax, 2  add bx, 3  mov dx, ax  mov ax, bx  mov bx, dx  dec cx  jmp **loop**  **out**: *;se afiseaza conținutul lui ax si bx* |

Procesorul ce execută acest cod rulează la frecvența de **1Ghz**. Metoda de *branch-prediction* folosită de fiecare este 1-bit counter (taken/not taken). Predictorul pornește cu o abordare pesimistă, counter-ul/state-machine-ul având starea inițială *not taken*. Predictorul este unul local, astfel că există un counter/state-machine pentru fiecare branch.

Se știu numărul de cicli de ceas pentru fiecare tip de instrucțiune: acces la memorie (**mov**) - 2 cicli, aritmetice/logice (**cmp**, **add**, **dec**) - 1 ciclu, de salt (**je**, **jmp**) - 40 cicli (dacă nu a fost prezis corect) sau 10 cicli (dacă a fost prezis corect).

Se știu **$a = 10**, **$b = 4**, **$c = 5**. Se cer următoarele:

1. Ce valori se afișează? **(1p)**
2. Să se evidențieze prin ce stări trec branch-predictor-ele semnalându-se și care branch-uri au fost prezise corect, respectiv incorect. **(4p)**
3. Să se calculeze timpul de rulare al programului. **(2p)**
4. Dacă ați putea îmbunătăți viteza de rulare a unei singure categorii de instrucțiuni (aritmetice/logice, acces la memorie sau de salt) astfel încât timpul de execuție al acelei categorii să scadă la jumătate, ce categorie ați alege pentru a obține cel mai bun speed-up în cazul programului vostru? Cât este acest speed-up? **(3p)**