Problemă examen CN

31.08.2016

Fie un procesorul ce rulează la frecvența de 1Ghz. Metoda de branch-prediction folosită este 2-bit counter (strongly taken/weakly taken/weakly not taken/strongly not taken). Predictor-ul pornește cu o abordare pesimistă, având starea inițială strongly not taken. Predictor-ul este unul local, astfel încât există un counter/state-machine pentru fiecare branch.

Fie următorul cod în assembly

mov ax, $a

mov bx, $b

mov cx, $c

mov dx, $d

**loop**: cmp cx, 0

je **out**

add ax, bx

inc bx

add dx, bx

add dx, ax

mov ax, bx

mov dx, ax

dec bx

dec cx

jmp **loop**

**out**: *; Se afiseaza conținutul din registrele ax, bx, cx si dx*

Se știu numărul de cicli de ceas pentru fiecare tip de instrucțiune

* Operațiile cu date și memorie (**mov**) durează 2 cicli
* Operațiile aritmetice/logice (**cmp**, **add**, **inc**, **dec**) durează 1 ciclu
* Operațiile de salt necondiționat (**jmp**) durează 10 cicli
* Operațiile de salt condiționat (**je**) durează 40 cicli (dacă nu a fost prezis corect) sau 10 cicli (dacă a fost prezis corect)

Se știu **$a = 1**, **$b = 20**, **$c = 3**, **$d = 0**. Se cer următoarele

1. **(1p)** Ce valori se afișează?
2. **(4p)** Evidențiați prin ce stări trec branch-predictor-ele indicând și care branch-uri au fost prezise corect, respectiv incorect.
3. **(2p)** Calculați timpul de rulare al programului.
4. **(3p)** Dacă ați putea îmbunătăți viteza de rulare a unei singure categorii de instrucțiuni dintre cele enumerate mai sus astfel încât timpul de execuție al acelei categorii să scadă la jumătate (nu poate scădea sub 1 ciclu), ce categorie ați alege pentru a obține cel mai bun speed-up în cazul programului vostru? Cât este acest speed-up?

Rezolvare problemă CN

31.08.2016

TODO

Problemă examen CN

01.09.2016

Tocmai v-ați angajat la o companie multinațională (pe un salariu considerabil) și prima voastră sarcină este să asigurați protecția celor mai noi tehnologii dezvoltate de companie, care sunt tinuțe într-un depozit secre. Pentru aceasta va trebui să contruiți un sistem care nu deschide usa depozitului decât dacă este introdus un cod corect. În pauza de prânz ați făcut deja majoritatea muncii, tot ce mai rămâne de făcut este logica de control.

Dispozitivul care controlează ușa este bazat pe FPGA și are 10 intrări de la 10 butoane (unul pentru fiecare cifră: **button0**, **button1**, ..., **button9**), o intrare de la un buton de **reset** și o ieșire **open\_door** (care deschide ușa). Scrieți un modul Verilog care să controleze acest dispozitiv și să aibă următoarea funcționalitate:

* Modulul așteaptă combinația corectă pentru a deschide usa (o succesiune de apăsări, nu mai multe în același timp)
* Starea inițială a modulului este: nu a fost apăsat niciun buton până acum și ușa este închisă
* Butonul **reset** aduce modulul în starea inițială oricând este apăsat
* Dacă un buton (**button0**, **button1**, …, **button9**, **reset**) este apăsat atunci intrarea corespunzătoare lui este 1, altfel este 0
* Dacă ieșirea (**open\_door**) este 0 atunci ușa este închisă, altfel este deschisă
* Dacă a fost introdus codul corect atunci ușa va ramane deschisă până la resetarea modulului
* Dacă nu a fost introdus codul corect, după apăsarea oricărei taste greșite ușa va rămâne închisă până la resetarea modulului
* Codul pentru deschiderea ușii este “123456” (adică trebuie apăsate, in ordinea asta, **button1**, **button2**, …, **button6**)

Exemplu de funcționare: se introduce “01547”, ușa nu se deschide; se introduce “12345”, ușa nu se deschide; se introduce “0123456”, ușa nu se deschide; se introduce “123456”, ușa se deschide; se introduce “123456789”, ușa se deschide.

Se cer

1. **(3p)** Graful stărilor, dacă automatul este de tip Mealy.
2. **(3p)** Graful stărilor, dacă automatul este de tip Moore.
3. **(4p)** Modulul Verilog, dacă automatul este de tip Moore.

Rezolvare problemă CN

01.09.2016

TODO