# Examen CN1 - Partea II - 09.06.2018

Nume: ………………………………………. Grupa: …………….

**1. (7p)** Fie următorul cod în assembly (x86):

0: mov eax, 0 *; eax = 0*

1: mov ecx, 4 *; ecx = 4*

2: xor edi, edi *; edi = 0*

**\_loop**:

3: cmp eax, ecx *; eax ? ecx*

4: je **\_out**

5: mov ebx, eax *; ebx = eax*

6: and ebx, 0x01 *; ebx &= 0x01*

7: cmp ebx, 1 *; ebx ? 1*

8: je **\_inc**

9: add eax, 1 *; eax += 1*

10: jmp **\_loop**

**\_inc:**

11: inc edi *; edi += 1*

12: add eax, 1 *; eax += 1*

13: jmp **\_loop**

**\_out**:

*; se afiseaza continutul lui edi*

Procesorul ce execută acest cod rulează la frecvența de **2Ghz**. Metoda de *branch-prediction* folosită de fiecare este **1-bit counter** (**taken/not taken**), starea inițială fiind ***not taken***. Predictorul este unul **local**, astfel încât există un state-machine pentru fiecare branch.

Știm 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, xor, and**) durează **1** ciclu, operațiile de salt necondiționat (**jmp**) durează **10** cicli, operațiile de salt condiționat (**jg, jl, je**) durează **50** cicli (dacă nu a fost prezis corect) sau **10** cicli (dacă a fost prezis corect).

Se cer următoarele:

1. **(1p)** Ce valoare afișează programul? Justificați!
2. **(5p)** Evidențiați prin ce stări trec branch-predictor-ele, semnalând și care branch-uri au fost prezise corect, respectiv incorect.
3. **(1p)** Care este timpul de rulare al programului?

**2. (3p)** Fie următorul cod în assembly (x86). Indentificați hazardele de date. Pentru fiecare menționați intrucțiunile implicate, arătând cauzele fiecărui hazard.

0: mov eax, ebx *; eax = ebx*

1: mov ebx, [0xCAFEBABE] *; ebx = MEM[0xCAFEBABE]*

2: add ecx, eax *; ecx += eax*

3: mov ecx, 0xCAFEBABE *; ecx = 0xCAFEBABE*

4: mov eax, ecx *; eax = ecx*

# Rezolvare

1.

1. 2, fiindca numara numerele impare in [0, 4)
2. je (linia 4): NT (corect), NT (corect), NT (corect), NT (corect), NT (gresit), T; je (linia 8): NT (corect), NT (gresit), T (gresit), NT (gresit), T
3. 322 cicli = 161 ns

2. I0 - I2: RAW (eax); I2 - I4: RAW (ecx); I3 - I4: RAW (ecx);

# Examen CN1 - Partea II - 09.06.2018

Nume: ………………………………………. Grupa: …………….

**1. (7p)** Fie următorul cod în assembly (x86):

0: mov eax, 4 *; eax = 4*

1: mov ecx, 0 *; ecx = 0*

2: xor esi, esi *; esi = 0*

**\_loop**:

3: cmp eax, ecx *; eax ? ecx*

4: je **\_out**

5: mov ebx, eax *; ebx = eax*

6: and ebx, 0x01 *; ebx &= 0x01*

7: cmp ebx, 0 *; ebx ? 0*

8: je **\_inc**

9: sub eax, 1 *; eax -= 1*

10: jmp **\_loop**

**\_inc:**

11: inc esi *; esi += 1*

12: sub eax, 1 *; eax -= 1*

13: jmp **\_loop**

**\_out**:

*; se afiseaza continutul lui esi*

Procesorul ce execută acest cod rulează la frecvența de **3Ghz**. Metoda de *branch-prediction* folosită de fiecare este **1-bit counter** (**taken/not taken**), starea inițială fiind ***not taken***. Predictorul este unul **local**, astfel încât există un state-machine pentru fiecare branch.

Știm 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, xor, and**), durează **1** ciclu, operațiile de salt necondiționat (**jmp**) durează **10** cicli, operațiile de salt condiționat (**jg, jl, je**) durează **60** cicli (dacă nu a fost prezis corect) sau **10** cicli (dacă a fost prezis corect).

Se cer următoarele:

1. **(1p)** Ce valoare afișează programul? Justificati!
2. **(5p)** Evidențiați prin ce stări trec branch-predictor-ele, semnalând și care branch-uri au fost prezise corect, respectiv incorect.
3. **(1p)** Care este timpul de rulare al programului?

**2. (3p)** Fie următorul cod în assembly (x86). Indentificați hazardele de date. Pentru fiecare menționați intrucțiunile implicate, arătând cauzele fiecărui hazard.

0: mov ebx, eax *; ebx = eax*

1: mov ebx,[0xDEADBEEF] *; ebx = MEM[0xDEADBEEF]*

2: add eax, ecx *; eax += ecx*

3: mov [0xDEADBEEF], eax *; MEM[0xDEADBEEF] = eax*

4: mov ecx, eax *; ecx = eax*

# Rezolvare

1.

1. 2, fiindca numara numerele pare in [4, 0)
2. je (linia 4): NT (corect), NT (corect), NT (corect), NT (corect), NT (gresit), T je (linia 8): NT (gresit), T (gresit), NT (gresit), T (gresit), NT
3. 412 cicli = 137.34 ps

2. I2 - I3: RAW (eax); I2 - I4: RAW (eax);