**SVEUČILIŠTE U SPLITU**

**FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE**

NAPREDNE ARHITEKTURE RAČUNALA (250)

2

**Toni Biuk**

**Tino Melvan**

**Split, svibanj 2018.**

**SADRŽAJ**

[1. 1](#_Toc514592790)

[2. 2](#_Toc514592791)

[3. 3](#_Toc514592792)

[4. 4](#_Toc514592793)

[5. 5](#_Toc514592794)

[6. 6](#_Toc514592795)

[7. Problem hardvera 7](#_Toc514592796)

[8. Prilog 9](#_Toc514592797)

[8.1. 9](#_Toc514592798)

[8.2. 9](#_Toc514592799)

[8.3. 9](#_Toc514592800)

[8.4. 9](#_Toc514592801)

[8.5. 9](#_Toc514592802)

[8.6. 9](#_Toc514592803)

[8.7. Primjer 7 9](#_Toc514592804)

# 

# 

# 

# 

# 

# 

# Problem hardvera

Ovaj primjer trebao bi demonstrirati čudno ponašanje brze memorije kao posljedica ovisnosti o drugom hardveru. Iako u potpunosti znamo kako bi se brza memorija trebala ponašati, činjenica je da je i dalje teško predvidjeti. Pogledom na primjer dan u prilogu (8.7), moglo bi se krivo zaključiti da će se funkcija koja inkrementira manje varijabli (WeirdnessAC) uvijek (ili barem u prosijeku) izvršavati brže od funkcije koja inkrementira više varijabli (WeirdnessABCD, WeirdnessACEG), naravno, ako su one po svemu ostalome identične. Puštanje sveukupnog programa otkriti će, pak, da to nije slučaj. Inkrement funkcija WeirdnessABCD i WeirdnessACEG uvijek su blizu, što se tiče vremena izvršavanja. Treća funkcija, WeirdnessAC, s druge strane, pomalo je nepredvidiva. Kolika puta se pojavila kao najbrža (Slika 7.1), toliko puta se pojavila i kao najsporija (Slika 7.2).



Slika 7.1. Brzi AC inkrement



Slika 7.2. Spori AC inkrement

Razlozi zašto bi ovo mogao biti slučaj su razni. Najlakše bi bilo zaključiti kako ovo ima veze s redoslijedom kojim instrukcije završe na procesoru, ili pak s činjenicom koliko se brzo otkrije podudaranje u brzoj memoriji. Problem može biti i dublji, ako kompajler (tj. procesor) koristi SSE (engl. *Streaming SIMD Extensions*) instrukcije. Takva vrsta ne radi s jednom po jednom varijablom, već ih kupi i do četiri odjednom, ukoliko su instrukcije dovoljno slične (što je svakako slučaj ovdje). No, da se radi o SSE instrukcijama, vremena bi bila mnogo sličnija nego što zapravo jesu.

# Prilog

## 

## 

## 

## 

## 

## 

## Primjer 7

#include <Utility/Clock.hpp>

#include <Utility/Traces.hpp>

#include <cstdint>

#include <cstdlib>

volatile uint32\_t A, B, C, D, E, F, G;

static void WeirdnessABCD() noexcept {

for (auto i = 0U; i < 200000000U; i++) {

A++;

B++;

C++;

D++;

}

}

static void WeirdnessACEG() noexcept {

for (auto i = 0U; i < 200000000U; i++) {

A++;

C++;

E++;

G++;

}

}

static void WeirdnessAC() noexcept {

for (auto i = 0U; i < 200000000U; i++) {

A++;

C++;

}

}

int32\_t main() noexcept {

Clock clock;

clock.Start();

WeirdnessABCD();

auto elapsedTimeMS = clock.ElapsedMiliSeconds();

INFO("A++; B++; C++; D++;\t\t%lld ms\n", elapsedTimeMS);

clock.Start();

WeirdnessACEG();

elapsedTimeMS = clock.ElapsedMiliSeconds();

INFO("A++; C++; E++; G++;\t\t%lld ms\n", elapsedTimeMS);

clock.Start();

WeirdnessAC();

elapsedTimeMS = clock.ElapsedMiliSeconds();

INFO("A++; C++;\t\t\t%lld ms\n", elapsedTimeMS);

return EXIT\_SUCCESS;

}