Generieke Asssembly Code Simulatie

21 april 2020

Student:
Folkert van Verseveld
11839341

Begeleiders:
Taco Walstra, Anthony van Inge, Edwin Steffens

Examinator:
Robert Belleman

Inhoudsopgave

Inleiding - Context

- Decennia aan onderzoek naar ontwerpen van geoptimaliseerde computerarchitecturen
- Afwegingen m.b.t. robuustheid, stabiliteit, veiligheid, processorsnelheid en stroomverbruik
- Statische en dynamische testprogrammatuur voor programmacorrectheid
- Probleem: verifiëren van programmacorrectheid van machine code programma's en compilers

Inleiding - Uitdagingen in programmatuur

- Meeste programmatuur geschreven in high-level programmeertalen (HLPL)
- HLPL gebruiken Compilers: complexe softwarepaketten.
- Praktijk: verificatie deels mogelijk; onhandig en tijdsintensief
- Imperfecties in bewezen correcte computerarchitectuur en specificatie: bijv. RISC-V instructieset, C11 compilermapping
- Doel: Simuleren van arbitraire machine code programma's voor diverse computerarchitecturen
- Uit literatuuronderzoek: geen adequate simulator

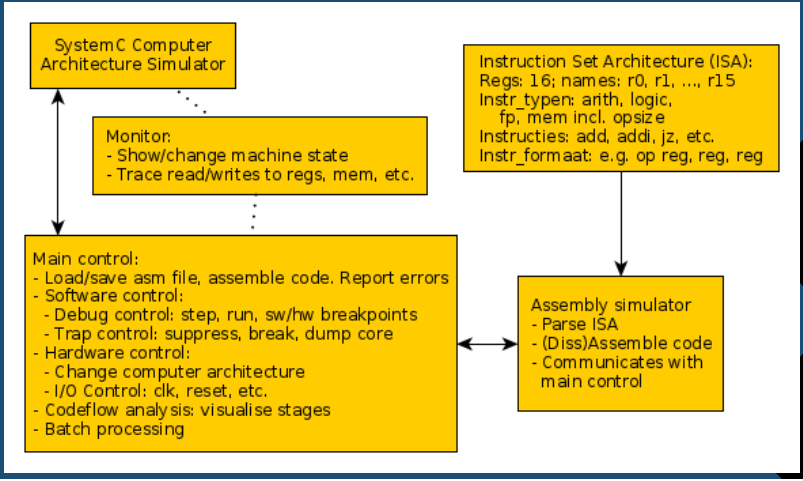
Inleiding - Oplossing en Aanpak

- Ontwerpen en implementeren van Generieke Assembly Code Simulator (GACS) om diverse computerarchitecturen te simuleren
  - Niet-triviaal: afwegingen m.b.t. performance, correctheid, onderhoudbaarheid en flexibiliteit
  - Steunend op Robins SystemC Computer Architecture Simulator
- Onderzoek: uitvoeren van optimalisatie-experimenten.
- Computerarchitecturen: in ieder geval subset van MIPS-I R3000 en een ARMv6/v7

Onderzoeksvraag

Welke metrieken en simulatiemodulen zijn noodzakelijk om een uitbreidbaar computerarchitectuur-onafhankelijk testframework te bieden om diverse computerarchitecturen te simuleren voor optimalisatie-experimenten van machine-code programma's?

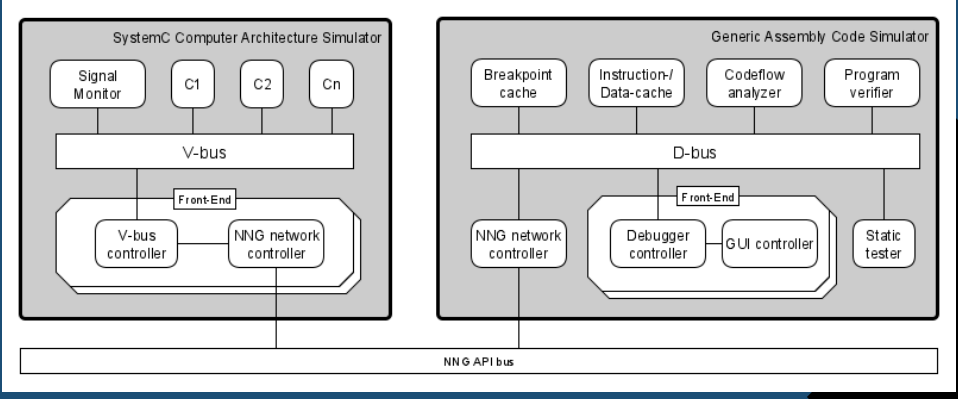
Ontwerp - Blokkenschema



Ontwerp - Eisen

- Cross-platform grafisch georiënteerde simulator
- Interactief of Batchprocessing
- Geïntegreerde debugger
- Pipeline stage visualisatie (indien aanwezig)
- Tools geïnspireerd door professionele Reverse-Engineering tools
  - Codeflow visualisatie
  - Gewenst: Opmerkingen plaatsen in code, data, stack e.d.

Ontwerp - Modulenschema



Onderzoek - optimalisatie-experimenten

- Codeflow, totale performance (CPI)
- Aantal gebruikte registers, maximale stackdiepte
- Aanpakken data-hazards: op welk niveau en waar het effectiefst?

Implementatievoortgang

- Opgezette libraries: SDL2, SystemC, NNGPP, JSONCPP, Flex, Bison
- Eenvoudige netwerkkoppeling met NNG en systeemkoppeling met SystemC
- Tijdelijk simpele computerarchitectuur
- Geïntegreerd: lexer, parser, assembler en dissassembler

Bedankt voor jullie aandacht

Subvragen

- Wat voor metrieken benodigt de simulator voor optimalisatie-experimenten?
- Welke simulatiemodulen zijn noodzakelijk om een qualitief onderzoek uit te voeren voor geoptimaliseerde computerarchitecturen?
- Hoe verschaft een simulator inzicht in de processen die gepaard gaan met statische en dynamische tests?
- Hoe kan GACS geïntegreerd of gekoppeld worden met een computerarchitectuursimulator?
- Met welke metrieken kan de code-flow geanalyseerd worden?

Simpele Computerarchitectuur

- Turing-compleet
- Non-pipelined 8 bit computer: 8-bits wordsize, 16 registers
- Programma ROM: 256 words van 20 bits
- Werkgeheugen: RAM van 256 words van 8 bits
- Wiskundig: Add, Overdracht: Move, Flow: jump on zero

Simpele Computerarchitectuur

Instruction Set Architecture:

Pseudocode	syntax	voorbeeld
r[dst] := r[lhs] + r[hrs]	ADD dst, lhs, rhs	ADD r3, r0, r1
r[dst] := r[lhs] + i	ADDI dst, lhs, i	ADDI r8, r2, 5
(*)mem[lhs] := r[rhs]	MOVE lhs, rhs	MOVE 4, r0
if z=1: goto a	JZ a	JZ stop

z: zero flag. 1 wanneer ADD of ADDI resulteert in 0. (*) operanden kunnen (r,r), (m,r), (r,m) en (m,m) zijn