En enkelt-cyklus mikroarkitektur

Dat1E godkendelsesopgave 2

Rune Andresen og Martin Zachariasen

16. oktober 2001

1 Formalia

Dette er den anden af to godkendelsesopgaver på kurset Dat1E, efterår 2001. Opgaven skal løses i grupper på indtil 3 personer. Opgaven bliver stillet mandag d. 22/10 2001 kl. 14.00 og skal afleveres i førstedelsadministrationen i 2 eksemplarer inden **mandag d. 29/10 2001 kl. 11.30**. Bedømmelserne foreligger i førstedelsadministrationen senest mandag d. 12/11 2001. I tilfælde af at opgaven ikke bliver godkendt i første omgang er genafleveringsfristen fredag d. 16/11 2001 kl. 11.30.

2 Formål

Formålet med opgaven er at opnå erfaring med brug af C++ og SimSys med henblik på anvendelse til løsning af K1, samt at konsolidere indsigt i digitallogik og talrepræsentation.

Der skal implementeres en enkelt-cyklus mikroarkitektur der kan udføre en udvalgt delmængde af MIPS' maskinsprog.

3 Om opgaven

3.1 Maskinsprog

Den implementerede enkelt-cyklus mikroarkitektur skal kunne udføre følgende MIPS maskinkodeordrer:

Kategori	Ordrer
R-type	ADD, SUB, SLT, AND, OR
Hop	BEQ, BNE, JR, JAL, J
I-type	ADDI, SLTI, ANDI, ORI
Lagertilgang	LW, SW
Særligt	STOP

Ordren STOP er ikke en rigtig MIPS ordre, men en ordre tilføjet til brug i opgaverne her på Dat1E. Den indkodes som FC000000 hexadecimalt. Indkodningen af de øvrige ordrer fremgår af COD [1], s. 153 og Appendix A. Udførelse af STOP ordren afslutter simulationen.

Der skal ikke undersøges for overløb (i forbindelse med ADD, SUB eller ADDI) eller genereres *exceptions*. Dog skal SLT og SLTI altid fungere korrekt.

Bemærk at ordrerne ADDI og SLTI anvender fortegnsforlængelse af den indlejrede 16-bit konstant, mens ANDI og ORI anvender nul-forlængelse.

3.2 Mikroarkitektur

Der skal implementeres en enkelt-cyklus mikroarkitektur som beskrevet i COD afsnit 5.1 til 5.3. Især henledes opmærksomheden på at figur 5.29 s. 372 i COD kan anvendes som udgangspunkt. Bemærk at I-type ordrerne samt BNE, JR og JAL ikke kan udføres af denne mikroarkitektur, hvorfor den må udvides til også at kunne håndtere disse.

Til implementationen skal simulationssystemet SimSys [2] anvendes. Installationen af SimSys på DIKUs førstedelssystem er beskrevet på

`http://www.diku.dk/teaching/2001e/dat1e/SimSys/inst.html`

Som udgangspunkt for udvikling af simulatoren er en skabelon stillet til rådigned på `~dat1e/G2/g2-skabelon.cc`.

Byggeklodsen `IdealMemIOSys` skal anvendes til data- og programlager. Konfigurer byggeklodsen til at have 1 Mb lager med en tilgangstid på 10 ns. Simulatoren skal kunne læse heltal fra standard input (*cin*) og skrive til standard output (*cout*). Udførelse af en LW ordre der læser fra adresse FFFFFFF0 skal føre til indlæsning fra *cin*, mens udførelse af SW ordre der skriver til adresse FFFFFFF4 skal føre til udskrivning til *cout*. Indlæsning og udskrivning af heltal foretages let med objekter af typen IntIO (se kursusbog 3 [2], s. 135–137). Den udleverede skabelon initialiserer lager og I/O i overensstemmelse med disse krav.

Den aritmetisk-logiske enhed fra SimSys opgave 3.6 passer til den krævede mikroarkitektur. Desuden anbefales det, at der udvikles en afkoder som beskrevet i SimSys opgaverne 3.7-3.9.

3.3 Værktøj og testprogrammer

Til hjælp til besvarelse af opgaven stilles følgende faciliteter (udover SimSys) til rådighed:

- *masm*, en makroassembler. *masm* oversætter fra symbolsk maskinsprog (filer med suffix "asm") til et format kaldet hex-kode (filer med suffix "hex").
- *mips*, en reference simulator for en delmængde af MIPS maskinsproget. *mips* læser og udfører filer genereret med *masm*. Endvidere kan *mips* bruges til at lave filer med spor af udførelsen. Disse filer kan så bruges af SimSys til at validere udførelsen.
- Kildetekster indeholdende en primitiv symbolsk disassembler for en delmængde af MIPS maskinsproget.

Disse faciliteter er nærmere beskrevet på

`http://www.diku.dk/teaching/2001e/dat1e/SimSys/mips.html`

Til brug for indkøring udleveres nogle testprogrammer der skal oversættes og simuleres med de udleverede værktøjer. Testprogrammerne er placeret i kataloget `~dat1e/G2`.

4 Krav til besvarelsen

Der skal afleveres en kort beskrivelse af løsningen (1–2 sider), alt udviklet kildetekst samt udskrift af kørsler af de udleverede testprogrammer. Desuden skal alle kildetekstfiler — herunder en README-fil der beskriver hvordan filerne oversættes til et færdigt program — gøres tilgængelige på en 1. delskonto (`di`-konto) ejet af en af deltagerne; filerne lægges i et katalog med navnet `G21E` og må ikke rettes efter rapportens aflevering.

5 Gode råd

Kurset har sin egen nyhedsgruppe, `diku.dat1e`, der er beregnet som elektronisk diskussionsforum for problemstillinger knyttet til kurset. Instruktorer og lærere læser indlæg i denne gruppe regelmæssigt, og bidrager gerne med opklaring af spørgsmål m.v. Brug nyhedsgruppen.

Der afholdes en spørgetime om G2 og SimSys onsdag d. 24/10 2001 kl. 11.15 i Aud. 3, HCØ.

I tilknytning til SimSys vedligeholdes en OSS (Ofte Stillede Spørgsmål), og en fortegnelse over rettelser til kursusbog 3. Disse sider opdateres efter behov og bør besøges hyppigt, især i rapportperioderne. Fejl i SimSys eller kursusbogen bedes indberettet som beskrevet på OSS-siden; andre spørgsmål kan bedre stilles i nyhedsgruppen.

Det anbefales at man anvender spor-faciliteten i SimSys, som er beskrevet i kursusbog 3, appendix A (se også SimSys øvelse 5.2). Skrivning til registrene kan valideres ved brug af byggeklodsen `ValidateWrite` (kursusbog 3, s. 168). Skrivning til lager og I/O vil automatisk blive valideret af `IdealMemIOSys` byggeklodsen, hvis spor-faciliteten anvendes. Bemærk at hvis spor-faciliteten anvendes, vil SimSys ikke udføre I/O som beskrevet i afsnit 3.2 (dvs. læse fra tastaturet og skrive til skærmen), men i stedet foretage indlæsning fra spor-filen, og validere udskrivning ved hjælp af spor-filen.

Til sidst nogle gode råd angående rapporten:

- Husk sidetal på alle sider — også bilag.
- Husk en indholdsfortegnelse som også refererer til bilag.
- Husk at referere til alle bilag fra selve rapporten.

Litteratur

[1] David A. Patterson and John L. Hennessy: Computer Organization and Design, 2nd ed., Morgan Kauffman 1998. ISBN 1-55860-491-X.

[2] Finn Schiermer Andersen: SimSys: A Framework for Digital Logic Simulation, Dat1E Kursusbog 3, DIKU 2001.