

G2
Maskinarkitektur
Efterår 2011

Jens Fredskov
Naja Mottelson
Søren Pilgård

2. oktober 2011

Indhold

1	Indledning	3
2	g2-1	3
3	addu, subu, and, or, nor, slt, jr	3
4	sll, srl, sra	4
5	addiu, andi, ori, slti	4
6	j, jal	4
7	g2-2	4

1 Indledning

Denne rapport dokumenterer gruppens arbejde med anden godkendelsesopgave i kurset Maskinarkitektur. Den indeholder, for såvidt vi kan se, korrekte besvarelser af begge underopgaver.

2 g2-1

Vores implementation af enkeltcykluskredsløbet følger, i store træk, samme mønster som den beskrevet i lærebogens. Udover at forlænge ALUop-wiren fra kontrollen til ALUKontrollen med 1 bit har vi dog tilføjet tre nye kontrolbits, som kort beskrives i figur 1

Navn	Effekt når lav	Effekt når høj
SignExtend	immediate-feltet fortegnudvides 16 pladser.	immediate-feltet nuludvides 16 pladser.
j	Ingen	PC'en sættes til $(PC+4[31:28], \text{address}, 2'b0)$
jal	Ingen	Register 31 sættes $PC+4$

Figur 1: Nye kontrolbits

De instruktioner der gennemgås grundigt i lærebogen (lw, sw, beq) har vi implementeret efter samme algoritme der præsenteres dér. Vi vil derfor ikke bruge yderligere tid på at gennemgå dem her.

3 addu, subu, and, or, nor, slt, jr

Alle R-instruktioner har vi implementeret på grundlæggende samme måde: I kontrollen sættes Write-registeradressen til rd (vha. RegDst) og RegWrite sættes så at det er muligt at skrive til registre. Resten af kontrollogikken foretages i ALU-kontrollen som modtager en 3-bit wire indeholdende ALUop-signalet fra kontrollen samt de 6 bits der hentes ud fra operationens funct-felt. Ud fra dette input genererer ALU-kontrollen et output på 6 bits, hvoraf de fire mindst betydende sendes til ALU'en som signal om hvilken af de aritmetisk-logiske instruktioner den skal udføre, hvorefter outputtet fra ALU'en skrives til registerbanken. I tilfældet af de grundlæggende R-instruktioner benyttes de to sidste bits i ALU-kontrollens output ikke.

I tilfælde af instruktionen jr benyttes ALU'en ikke. Det eneste kredsløbet foretager sig ved denne instruktion er at sende et signal til en MUX der vælger imellem at skrive den standard-inkrementerede adresse til PC'en ($PC+4$ i dette tilfælde) eller outputtet fra register rs.

4 sll, srl, sra

Da skifteoperationerne ligeledes er R-instruktioner er de implementeret med samme kontrollogik som ovenfor. Her sættes den fjerde bit i ALU-kontrollens output dog, og gives som selector-input til en MUX der giver shamt som den første operand til ALU'en i stedet for register rs.

5 addiu, andi, ori, sli

I-instruktionerne er implementeret på samme facon som R-instruktionerne med de undtagelser at der skrives til register rt samt at ALUsrc-bitten sættes i kontrollen. ALUsrc gives som selector-input til en MUX som vælger hvilket input der gives til ALU'ens anden operand. Når ALUsrc er sat vælges operationens immediate-felt, som er blevet enten nul- eller fortegnsudvidet afhængigt af instruktionen. Nul- eller fortegnsudvidelse specificeres ved kontrollens SignExtend-bit.

6 j, jal

Som antydnet i indledningen er J-instruktionerne implementeret med deres 'egne' bits i kontrollen, hhv. j og jal. Når j-instruktionen udføres er denne bit den eneste der er sat. Signalet herfra går til en MUX der vælger imellem at skrive den standard-inkrementerede adresse til PC'en eller den der specificeres af instruktionens adressefelt.

I tilfældet af instruktionen jal sættes RegWrite, så at der kan skrives til registerbanken, samt både j- og jal-bitten, hvis effekt er at skrive den standard-inkrementerede PC til register 31. *Her føres rd- og rt-registrene igennem en dobbelt-MUX, og hvad der foregår her kan Naja ikke rigtig lure. Hilfe!.*

7 g2-2

Vores implementation af underopgave 2 er at finde i den vedlagte fil g2-2.asm. Som dokumentation for denne henviser vi til de kommentarer der er anført i kildekodefilen selv.