Processoren 2014

Week 4: De von Neumann architectuur

Uiterste inleverdatum: 8 december

Lever de volgende opgaven in via email naar je studentassistent. Zorg dat [Proc]Week4 in het subject van je email staat zodat we die niet over het hoofd kunnen zien.

Opmerking 1: Deze keer zijn het niet zoveel opgaven maar wel enige die het nodige leeswerk vergen.

Opmerking 2: Zowel op de pagina van de practicum-processor en in opgave 2 en 3 wordt gebruik gemaakt van het begrip instructieformaat. Normaal bestaat een instructie van een processor uit een opcode (operation code) en een of andere manier om de operanden van die instructie aan te duiden. Je kunt de instructies van deze processor dus groeperen naar hun manier van operandenduiding. Ze delen dan een bepaald instructieformaat oftewel deze specifieke manier van opcode met duiding van de operanden.

1) Lees de beschrijving van de practicum-processor op

   `https://lab.cs.ru.nl/algemeen/Processoren/Practicum`.

   Geef een logisch overzicht over de verbindingen tussen ALU en registers die nodig zijn om alle formaat 4 rekeninstructies uit te voeren b.v. in een diagram naar analogie van slide 12 van het college.

   Geef aan op welke punten er, afhankelijk van de uit te voeren berekening, een keuze gemaakt moet worden.

   Welke controle signalen moet de besturingseenheid van je processor genereren om alles in goede banen te leiden? Van welk deel van de instructie zijn die afkomstig?

2) De MIPS processor is een processor die gebruik maakt van 32 bits instructies. Deze processor heeft 32 algemene registers die dus met 5 bits gecodeerd kunnen worden. Aan de bovenste 6 bits van de instructie kan de MIPS zien welk formaat een instructie heeft:

Formaat	31..26	25..21	20..16	15..11	10..6	5..0
R	opcode	Rs	Rt	Rd	shift	functie
I	opcode	Rs	Rt	16 bits constante		
J	opcode	26 bits adres				

   In de bovengegeven tabel coderen Rs, Rt en Rd voor registers. Er zijn maar 2 instructies n.l. J (Jump; met opcode 2) en JAL (Jump and Link; met opcode 3) die het J-formaat hebben. Bij de R-instructies bepalen de onderste 6 bitten (het functie-veld) mede welke instructie uitgevoerd wordt. Stel dat er m instructies zijn met het I-formaat, hoeveel mogelijke instructies zijn er dan met het R-formaat? (Note: het antwoord zal je misschien verbazen en er zijn dan ook heel veel "witte plekken" in de MIPS instructieset).

3) Een processor-ontwerper wil een machinetaal met machinecodes van 16 bits ontwerpen en vraagt zich af hoe hij de instructieformaten het beste kan verdelen. Als hij de processor n instructies met twee operanden (bv. READ) geeft, hoeveel instructies met één operand (bv. PUSH, POP) kan de machinetaal maximaal hebben? Om één operand te beschrijven zijn 6 bits nodig (tentamen 2 mei 2012).

Facultatief

Na het doorlezen bij onderdeel 1 van deze opgavenset, zou je ook in staat moeten zijn om de adder/substractor die je al eerder had gemaakt uit te breiden naar een algemenere ALU die alle formaat4 instructies aankan. Deze veralgemeende ALU krijgt dus een 3 bit opcode signaal gevoerd die bepaalt welke operatie hij zou moeten uitvoeren. Voor alle logische operatoren kun je in rtlib.logic de benodigde componenten vinden. Bedenk ook dat je voor de implementatie van AND en BIC de conditionele inversie die je ook al voor de aftrekker had kan hergebruiken. Als je nu aan je testbench een extra Ipinvector voor de opcode toevoegt kun je alle nieuwe functionaliteit testen. Bedenk ook hoe je nu op basis van de opcode de output van de ALU moet bepalen (en er ligt hier een heel logisch ontwerp voor de hand).

Ook dit onderdeel is niet verplicht, maar door nu al met Hades en de practicum processor te starten vermijd je haastige spoed aan het einde van het trimester. Jullie studentassistent kan je natuurlijk wel altijd om feedback vragen.