Processoren 2014

Week 8: Oude opgaven en tentamenvragen

1) Converteer naar binaire 8-bits-getallen in twee-complement en bereken zoals in de practicum-processor (maar dan met 8 bits brede data):

   a) **(1 punt)** $-21_{\text{dec}} + 59_{\text{dec}}$

   b) **(1 punt)** 77_{dec} **xor** -25_{dec}

   c) **(1 punt)** $-12_{\text{dec}} - 120_{\text{dec}}$

   d) **(1 punt)** 102_{dec} **and** 21_{dec}

   e) **(1 punt)** 23_{dec} naar links geroteerd om 5 bits

   (1 punt) Geef voor elk van de bovenstaande berekeningen aan op welke waarden de vlaggen gezet worden (volgens de regels van de practicum-processor, maar dan voor 8-bits-getallen). (24 januari 2014)

2) Sommige processoren voeren alle berekeningen met hulp van de stack uit. Een rekeninstructie leest de bovenste twee elementen van de stack, voert de betreffende operatie uit en slaat het resultaat weer op de stack op. Het voordeel is dat een rekenoperatie geen enkele operand meer hoeft op te geven.

   Schrijf een programma voor een dergelijke stapel-processor dat berekent:

   $$X := (A - B \times C)/(D + E)$$

   Toegestane instructies zijn:

READ	*variabele*	lees een variabele en push haar waarde op de stack
WRITE	*variabele*	pop het bovenste element van de stack en sla het in de variabele op
ADD		tel de twee bovenste elementen van de stack bij elkaar op
SUB		trek het bovenste element van de stack af van het tweede element
MUL		vermenigvuldig de twee bovenste elementen van de stack
DIV		deel het tweede element van de stack door het bovenste element

3) Een processor-ontwerper wil een machinetaal met machinecodes van 32 bits ontwerpen en vraagt zich af hoe hij de instructieformaten het beste kan verdelen. Als hij de processor n instructies met twee operanden geeft, hoeveel instructies met drie operanden kan de machinetaal maximaal hebben? Om één operand te beschrijven zijn 6 bits nodig. (Variant van 31-1-2012).

4) We geven eerst een licht gewijzigde formulering van deze vraag die vorig jaar gesteld is op het tentamen van 4 maart 2014.

   > Schrijf een programma in practicum assembly dat het kleinste positieve (i.e. > 0) element van een array bepaalt. Hierbij staat op locatie array het eerste array element, op locatie array+4 het tweede, etc. De locatie

nrelts bevat het aantal elementen van het array. De elementen van het array zijn één machinewoord groot (dwz. 32 bits) en moeten signed behandeld worden. Dit programma zal dus als volgt eruit zien.

```
        .START  0
        ...                     # dit gedeelte moet je zelf schrijven
        WRITE   ..., [result]
        HALT
result: .DATA   0
nrelts: .DATA   ...             # Aantal elementen van het array
array:  .DATA   ...             # Het eerste element van het array
        .DATA   ...             # Het tweede element van het array
        ...                     # Et Cetera
```

a) Formuleer eerst in pseudo C, C++ of Java een algoritme dat hetzelfde doet.

b) Formuleer nu aan de hand van je beschrijving van onderdeel a het programma in practicum assembly.

5) Stel dat we de practicum processor willen uitrusten met een pipeline waarbij we de fetch, decode, ALU en (data) memory fase van de instructie elk een pipelinestage geven (zoals ook op het laatste college besproken) en dat we daartoe instructie en data ram interfaces scheiden (anders kunnen ze niet in parallel werken).

Afhankelijkheden tussen de pipelinestages kunnen zonder maatregelen tot foutief gedrag leiden, de zgn. hazards. Hazards kunnen echter op verschillende manieren ontstaan. Geef (zoveel mogelijk) verschillende soorten van hazards, waarbij je per hazard met een twee- of drieregelig voorbeeld in practicum assembly en een kort commentaar het probleem aangeeft. (licht gewijzigde versie van 24 januari 2014. We hebben dit afgelopen maandag wat uitgebreider besproken en geshowd dan vorig jaar ;-).