Assembleertaal

# Registers

In het boek wordt een aantal registers van de CPU genoemd. Behalve de registers R0 tot en met R9 waarin gegevens tijdelijk worden opgeslagen, zijn dat:

* De **programcounter** (**PC**): hierin staat het geheugenadres van de *volgende* instructie die verwerkt gaat worden;
* Het **instructieregister** (**IR**): hierin wordt door de control-unit de instructie gezet, die vervolgens wordt geanalyseerd en uitgevoerd;
* Het **statusregister** (**SR**): elke bit in dit register is een zogenaamde vlag (flag): een bit dat op 1 gezet wordt als er een bepaalde gebeurtenis optreedt tijdens het uitvoeren van een instructie. Het bit is 0 als die gebeurtenis niet optreedt.

Zo zijn er:

* + De **zero-flag (ZF**): dit bit wordt op 1 gezet zodra het resultaat van een berekening nul oplevert. Bijvoorbeeld een aftrekking of vergelijking van twee dezelfde getallen.
  + De **negatieve-flag (NF**); wordt op 1 gezet als het resultaat van een berekening of vergelijk negatief is
  + Voorbeeld: CMP R0 #12 JNG

Het resultaat van de **CoM**pare-instructie is dat de negatieve-flag (NF) op 1 gezet wordt als R0 – 12 een negatief getal oplevert. De instrucite **JNH (Jump Not Greater)** kijkt vervolgens of de NF op 1 of op 0 staat. Staat hij op 1 dan wordt gesprongen, anders niet. **JNG** kan dan ook gelezen worden als Jump If Negative.

# Uitleg van een programma

We bespreken nu hoe een eenvoudig machinetaalprogramma door de CPU verwerkt wordt. Bedenk dat in het geheugen uitsluitend nullen en enen staan. Voor het leesgemak hebben we de instructies geschreven in assemblertaal en de getallen decimaal opgeschreven.

Het programma staat in het werkgeheugen op de geheugenplaatsen met de adressen 1000 tot en 1010.

Verder staat er in het werkgeheugen op de plaatsen 2000 en 2001 twee getallen, 2002 is gereserveerd voor het resultaat.

De opbouw doen we in Excel.

??? betekent dat de inhoud van de betreffende geheugenplaats of register onbekend is.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **REGISTERS CPU** |  |  |  |  |  |  |  |  | **GEHEUGEN** |  |  |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |  |  |  |  |  |
| 1000 | ??? | ??? | ??? | ??? | ??? | 0 |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | **ADRES** | **DATA** |  |
|  |  |  |  |  |  |  |  |  | 1000 | LOAD R1 2000 | |
|  |  |  |  |  |  |  |  |  | 1001 | LOAD R2 2001 | |
|  |  |  |  |  |  |  |  |  | 1002 | LOAD R3 #0 | |
|  |  |  |  |  |  |  |  |  | 1003 | LOAD R4 #0 | |
|  |  |  |  |  |  |  |  |  | 1004 | CMP R2 R4 | |
|  |  |  |  |  |  |  |  |  | 1005 | JEQ 1009 | |
|  |  |  |  |  |  |  |  |  | 1006 | ADD R3 R3 R1 | |
|  |  |  |  |  |  |  |  |  | 1007 | ADD R4 R4 #1 | |
|  |  |  |  |  |  |  |  |  | 1008 | JMP 1004 | |
|  |  |  |  |  |  |  |  |  | 1009 | LOAD 2002 R3 | |
|  |  |  |  |  |  |  |  |  | 1010 | END | |
|  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | 2000 | 123 |  |
|  |  |  |  |  |  |  |  |  | 2001 | 5 |  |
|  |  |  |  |  |  |  |  |  | 2002 | ??? |  |
|  |  |  |  |  |  |  |  |  | 2003 | ??? |  |

Zodra dit programma gestart wordt, plaatst het besturingssysteem 1000 in de programcounter (PC) van de CPU.

**Toestand registers:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |
| 1000 | ??? | ??? | ??? | ??? | ??? | 0 |

De zero-flag in het statusregister (SR) : de rest van het SR hebben we hier niet nodig Vervolgens wordt één instructiecyclus uitgevoerd:

1. De control unit van de CPU haalt de instructie op die op de geheugenplaats staat die in de PC wordt aangewezen (= 1000) en plaatst die in het instructieregister. De PC wordt met 1 opgehoogd.
2. De instructie wordt geanalyseerd.
3. De instructie **LOAD R1 2000** wordt uitgevoerd: in register 1 wordt **123** gezet (de inhoud van geheugenplaats 1000).

**Toestand registers:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |
| 1001 | LOAD R1 2000 | **123** | ??? | ??? | ??? | 0 |

De volgende instructie wordt uitgevoerd:

1. De instructie op 1001 wordt opgehaald en de PC wordt 1002
2. De analyse van de instructie
3. De instructie LOAD R@ 2001 wordt uitgevoerd (in R2 komt nu 5 te staan)

**Toestand registers:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |
| 1002 | LOAD R2 2001 | 123 | **5** | ??? | ??? | 0 |

Nadat de volgende twee instructies opgehaald, geanalyseerd en uitgevoerd zijn, zien de registers er als volgt uit:

**Toestand registers:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |
| 1004 | LOAD R4 #0 | 123 | 5 | **0** | **0** | 0 |

De volgende instructie: **CMP R2 R4 (CoMPare** of vergelijk **R2** met **R4**) doet het volgende:

De inhoud van de registers R2 en R4 worden met elkaar vergeleken als de inhoud gelijk is, wordt in het status register (SR) de ZF op 1 gezet en anders op 0.

**Toestand registers:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |
| 1005 | CMP R2 R4 | 123 | 5 | 0 | 0 | **0** |

Bij de volgende instructie: **JEQ 1009** ( Jump if **EQ**ual: spring indien gelijk) wordt naar de inhoud van de ZF gekeken.

Als daar een 0 staat, gebeurt er niets en is de volgende instructie die opgehaald wordt, die welke op geheugenplaats 1006 staat. Als de inhoud echter een 1 is, wordt de inhoud van de PC vervangen door 1009, zodat de volgende instructie welke opgehaald wordt die van geheugenplaats 1009 is.

In dit geval is de ZF 0 en blijft de PC dus op 1006 staan.

**Toestand registers:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |
| 1006 | JEQ 1009 | 123 | 5 | 0 | 0 | 0 |

**ADD R3 R3 R1**

Tel de inhoud van R3 en R1 bij elkaar op en zet het resultaat weer terug in R3.

(Dat komt eigenlijk neer op het volgende: tel bij de inho8ud van R3 R1 op: 0 + 123 = 123) **Toestand registers:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |
| 1007 | ADD R3 R3 R1 | 123 | 5 | **123** | 0 | 0 |
| **ADD R4 R4 #1** betekent dat R4 met 1 opgehoogd wordt.  **Toestand registers:** | | | |  |  |  |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |
| 1008 | ADD R4 R4 #1 | 123 | 5 | 123 | **1** | 0 |

**JMP 1004**

De PC heeft direct na het ophalen van deze instructie (**J**u**MP**) even op 1009 gestaan, maar wordt tijdens het uitvoeren van deze opdracht op 1004 gezet.

Het resultaat is dat na deze instructie de PC op 1004 staat, zodat **CMP R2 R4** nu voor de tweede keer uitgevoerd zal worden.

**Toestand registers:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |
| 1004 | JMP 1004 | 123 | 5 | 123 | 1 | 0 |
| registers als volgt:  **Toestand registers:** | |  | Nadat de instructies 1004 tot en met 1007 voor de tweede keer uitgevoerd zijn, is de toestand van de |  |  |  |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |
| 1004 | JMP 1004 | 123 | 5 | **246** | **1** | 0 |

De instructies 1004 tot en met 1007 zullen nog drie keer herhaald worden, waarna de toestand van de registers na de vijfde keer er als volgt uitziet:

**Toestand registers:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |
| 1004 | JMP 1004 | 123 | 5 | **615** | **5** | 0 |

Vervolgens wordt **CMP R2 R4** voor de zesde keer uitgevoerd, maar nu zijn de inhoud van Re en R4 aan elkaar gelijk en dus wordt de ZF op 1 gezet.

**Toestand registers:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |
| 1004 | JMP 1004 | 123 | 5 | 615 | 5 | **1** |

Nu gebeurt er bij de instructie **JEQ 1009** wat anders dan in de voorafgaande vijf keer. JEQ treft in de ZF een 1 aan en zal de inhoud van de PC (die intussen op 1006 staat) veranderen in 1009. Vervolgens wordt de zero-flag weer op 0 gezet.

**Toestand registers:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |
| 1009 | JEQ 1009 | 123 | 5 | 615 | 5 | **0** |

De instructies 1006 tot en met 1008 worden nu overgeslagen en de instructies op 1009: **LOAD 2002 R3** wordt opgehaald, geanalyseerd en uitgevoerd. De inhoud van R3 wordt op de geheugenplaats 2002 neergezet.

**Toestand registers:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |
| 1010 | LOAD 2002 R3 | 123 | 5 | 615 | 5 | 0 |

**Toestand werkgeheugen:**

|  |  |
| --- | --- |
| 2000 | 123 |
| 2001 | 5 |
| 2002 | 615 |

De laatste opdracht (**END)** vertelt het besturingssysteem dat het de zaak weer over kan nemen.

# Oefeningen

1. Onderzoek wat het volgende programma doet. Ga stap voor stap na hoe de inhoud van de registers steeds verandert. (werk het uit in Excel)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **REGISTERS CPU** |  |  |  |  |  |  |  | **GEHEUGEN** |  |  |
| **PC** | **IR** | **R1** | **R2** | **R3** | **R4** | **ZF** |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | **ADRES** | **DATA** |  |
|  |  |  |  |  |  |  |  | 1000 | LOAD R1 2000 | |
|  |  |  |  |  |  |  |  | 1001 | LOAD R2 #0 | |
|  |  |  |  |  |  |  |  | 1002 | LOAD R3 #0 | |
|  |  |  |  |  |  |  |  | 1003 | CMP R1 R3 | |
|  |  |  |  |  |  |  |  | 1004 | JEQ 1008 | |
|  |  |  |  |  |  |  |  | 1005 | ADD R2 R2 R1 | |
|  |  |  |  |  |  |  |  | 1006 | ADD R3 #1 | |
|  |  |  |  |  |  |  |  | 1007 | JMP 1003 | |
|  |  |  |  |  |  |  |  | 1008 | LOAD 2001 R2 | |
|  |  |  |  |  |  |  |  | 1009 | END | |
|  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | 2000 | 2 |  |
|  |  |  |  |  |  |  |  | 2001 | ??? |  |
|  |  |  |  |  |  |  |  | 2002 | ??? |  |
|  |  |  |  |  |  |  |  | 2003 | ??? |  |