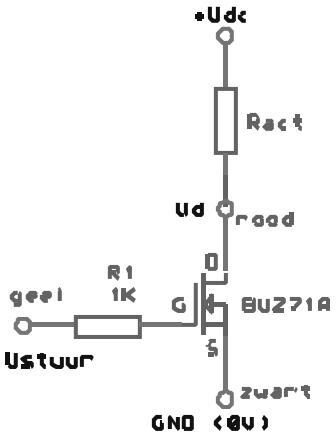
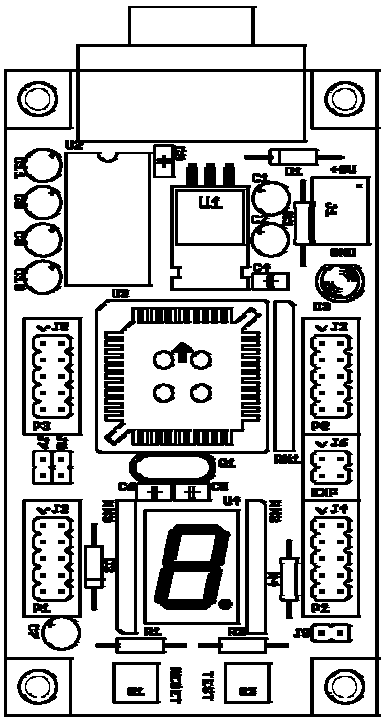
Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica
Practicumgroep, Elektrotechnisch Practicum

Practicumhandleiding

Elektronica voor IO

Code ET2-045ID





Redactie:
M. Schumacher
J.L.J.M. van Velzen

augustus 2006

Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica
Practicumgroep, Elektrotechnisch Practicum

Practicumhandleiding

Elektronica voor IO

Microcontroller-practicum

Code ET2 045ID

Redactie, coördinatie en instructies:
J.L.J.M. van Velzen (Cornelis Drebbelweg 5/Gebouw 35, kamer 01.110, tel: 85743)

Practicumzaal: zaal 01.140

augustus 2006

Practicum-info

Tijd:	morgen: 8.45 uur tot 12.30 uur middag: 13.30 uur tot 17.15 uur
Plaats:	Elektrotechnisch Practicum Cornelis Drebbelweg 5/Gebouw 35 Practicumzaal 01.140
Route:	De Cornelis Drebbelweg is rechts van het gebouw voor EWI. Gebouw 35 ligt aan de rechterkant. Trap op naar de eerste verdieping. Bovenaan naar links. De zaal is aan de rechterkant van de gang.
Kosten:	Op de eerste practicumdag moeten de benodigde computeronderdelen worden aangeschaft. De kosten zijn € 10.00, contant te voldoen.
Voorbereiding:	DOEN ! ! ! ! ! Bestudeer de onderwerpen 80C51 en assembly-programmering van de collegehandleiding en hoofdstuk 1 en 2 van deze practicumhandleiding.
Toelatingsvoorwaarden:	Aan de volgende regels zal strikt de hand worden gehouden. Een student die niet in het bezit is van eigen practicumhandleiding zal de toegang tot het practicum worden ontzegd. Een student die de practicumhandleiding niet blijkt te hebben bestudeerd zal van het practicum worden verwijderd. Bij verhindering met een geldige reden, bijvoorbeeld ziekte, moet u dit vooraf melden aan de practicumcoördinator. Afwezigheid zonder kennisgeving kan tot uitsluiting leiden.
Beoordeling:	Als na het practicum is gebleken dat u over voldoende kennis en vaardigheden beschikt, wordt een voldoende (geen cijfer) toegekend. Zonder een voldoende voor het practicum is het cijfer voor het vak Elektronica niet geldig.
Practicumcoördinatoren:	Voor het analoge deel: M.Schumacher@tudelft.nl, tel: 81850. Voor het digitale deel: J.L.J.M.vanVelzen@tudelft.nl, tel: 85743.

Inhoud van het practicum

o*chtend of middag 1:*
Het bouwen van de practicumcomputer.
Testen met een oscilloscoop.
Testen met een testprogramma.
Het schrijven van een zeer eenvoudig programma in hexadecimale vorm.
Het programmeren van de FLASH-EPROM.

ochtend of middag 2:
Het schrijven van een programma in assembly met behulp van een editor.
Het assembleren met een assembler.
Het programmeren van de FLASH-EPROM.

De opdrachten hebben betrekking op invoer en uitvoer, input met een drukschakelaar en output naar een 7-segment display.

ochtend of middag 3:
Het schrijven van een besturingsprogramma voor een aan te sluiten extern apparaat bijvoorbeeld een moto, een halogeenlamp of een luidspreker.

INHOUD

1 De hardware van de computer

1.1 Belangrijkste technische specificaties met de 89V51RD2 controller

Bruikbare microcontrollers uit de 8051 reeks
87C51 : 128 byte intern data RAM, 4 Kb intern EPROM
87C52 : 256 byte intern data RAM, 8 Kb intern EPROM
89C51Rx2 : 512 byte intern data RAM, intern Flash EPROM
89V51RD2 : 512 byte intern data RAM, 64 KB intern Flash EPROM
x = A: 8k Flash EPROM
x = B: 16k Flash EPROM
x = C: 32k Flash EPROM
x = D: 64k Flash EPROM

Deze typen zijn geschikt om op het 89V51-bordje te gebruiken. De uitvoering met flash-geheugen kan op het bordje worden geprogrammeerd, dus zonder gebruik te maken van een EPROM-programmer. Deze manier van programmeren heet ISP (In System Programming). De typen 80C31, 80C32, 80C51 en 80C52 zijn ongeschikt voor het bordje.
Op het practicum wordt de 89V51RD2 gebruikt.
De behuizing van deze microcontroller is 44 pens PLCC (Plastic Leadless Chip Carrier).

Extern geheugen
Er is geen extern programmageheugen op het bordje. Programma's worden in het flash-geheugen van de controller gezet. Extern data-geheugen kan zonodig worden aangesloten via de connectors J2 (poort P0), J4 (poort P2) en J6 (signalen ALE, RD# en WR#).

Jumper J7 en J8
J7 verbindt P3.0/RxD van de controller met pen 3 van J5 (P3.0)
J8 verbindt P3.1/TxD van de controller met pen 4 van J5 (P3.1)
Door deze jumpers weg te laten is het niet mogelijk om per ongeluk via J5 de seriële communicatie (microcontrollerpennen RxD/TxD) te verstoren.

Jumper J9
Behalve de 89V51RD2 kan ook een 89C51Rx2 worden "geflashed". Met J9 wordt PSEN via een weerstand van 2K2 met GND verbonden. Na RESET komt een 89C51Rx2 controller in de flash-mode. Het gebruik van een druktoets in plaats van een jumper is aan te bevelen.

8 bit bidirectionele datapoorten P0, P1, P2 en P3
Deze poorten zijn toegankelijk via de connectors J2, J3, J4 en J5.
P0 heeft externe pull-up weerstanden met waarde 10K. P1, P2 en P3 hebben interne pull-up weerstanden (90K).

Stroomverbruik
35 mA (exclusief 7-segment display en externe schakelingen).

Voedingsspanning
(zonder koelplaat op spanningsstabilisator 7805)
De maximale spanning is gebaseerd op een maximaal vermogen van 1 Watt dat in de spanningsregulator wordt opgenomen, waarbij de temperatuur 65 graden stijgt.
stroomverbruik 35 mA: 8 tot 35 Volt DC (35 Volt is het absolute maximum).
stroomverbruik 100 mA: 8 tot 18 Volt DC (7-segment display trekt stroom en/of externe hardware aangesloten).

1.2 Geheugen-organisatie

Met de 89V51RD2
0000 – FFFF : interne FLASH-EPROM (program memory)
00 – FF : 256 bytes standaard RAM (80C52 compatibel)
80 – FF : 128 SFR lokaties
00 – 2FF : 768 bytes externe RAM
(zet bit EXTRAM in SFR AUXR en gebruik MOVX instructie)

1.3 Input- en output-poorten

De standaard 8051 heeft vier 8-bit parallelle poorten (P0, P1, P2 en P3). Poorten zijn verbindingen met de buitenwereld. De vier poorten kunnen door een programma worden benaderd via de SFR's (Special Function Registers) P0, P1, P2 en P3.

Door in een programma een poort te lezen wordt het niveau van de elektrische spanningen op de uitwendige pennen van een poort in een register van de microcontroller ingelezen. Daarbij wordt een spanning tussen 2,5 en 5 Volt gezien als een logisch "1" en een spanning tussen 0 en 1 Volt als een logische "0". Pen 0 van een poort komt overeen met bit 0 van het SFR van die poort, enzovoort. Via poorten kunnen gegevens opgehaald worden uit een aangesloten randapparaat.

Voorbeeld:
kopieer de 8 bits van poort P1 in de accumulator (ofwel: lees P1)
machinecode : E5 90
assembly : MOV A, P1

Bij output wordt een byte in een poortadres geschreven. De spanning op een pen van de poort komt daarna overeen met het overeenkomstige bit in het poortgeheugen.

Voorbeeld:
stuur het bitpatroon in A naar poort P1 (ofwel: schrijf P1)
machinecode: F5 90
assembly: MOV P1,A

Ook bit-instructies zijn toe te passen op een poort. Zo kan men een enkel bit wijzigen zonder de overige pennen te beïnvloeden.

Voorbeeld:

C2 97	CLR	P1.7	maak pen 7 van poort P1 laag
A2 B2	MOV	C,P3.2	pen 2 van poort P3 naar de carry-vlag

In de practicumcomputer zijn alle poorten beschikbaar als 8 bits bidirectionele datapoort. Dit betekent dat via elke poortpen zowel input als output mogelijk is. De inwendige outputschakeling van P1, P2 en P3 bevat per bit een geschakelde transistor en een pull-up weerstand. Bij de vrijwel niet meer gebruikte N-MOS uitvoering 8051 is de weerstandswaarde van deze pull-up weerstand 9 Kohm, bij de C-MOS-uitvoering 80C51 is dat 90 Kohm. P0 heeft geen interne pull-up weerstanden. Deze zijn extern aangebracht.

Een logisch "0" op de output is hard (relatief laagohmige spanningsbron). Een "1" daarentegen gedraagt zich als een hoogohmige bron (de pull-up weerstand). Men dient rekening te houden met deze eigenschappen bij het aansluiten van een randapparaat. Bij input bepaalt het randapparaat het signaalniveau op de pennen van de poort. Dit is alleen mogelijk als output latch geen harde "0" bevat. Om de poortpen als input te kunnen gebruiken moet de pen hoogohmig zijn door dat er een "1" in de output staat.

Voorbeeld: MOV P1,#0FFH zet alle 8 bits van poort P1 op input

Bij een reset van de controller (bij het inschakelen van de voedingsspanning en met drukknop RESET) wordt de waarde FFH in de poorten gezet. Dus alle poorten zijn na reset automatisch geschikt voor input (en uiteraard ook output).

De poorten P0, P1, P2 en P3 van de controller zijn op het bordje toegankelijk via connectors met een gelijknamige aanduiding.

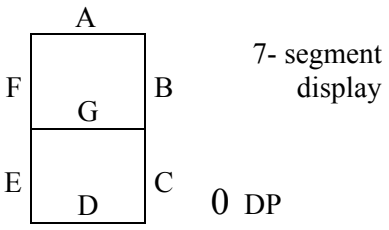
connector P0

pen	bit	functie
1		GND (aarde = 0 Volt)
2		VCC (voedingsspanning = 5 Volt)
3	P0.0	i/o, D0/A0
4	P0.1	i/o, D1/A1
5	P0.2	i/o, D2/A2
6	P0.3	i/o, D3/A3
7	P0.4	i/o, D4/A4
8	P0.5	i/o, D5/A5
9	P0.6	i/o, D6/A6
10	P0.7	i/o, D7/A7

Poort P0 is een bidirectionele datapoort met "open drain" uitgangen. Op het bordje zijn pull-up weerstanden aangebracht. Deze kunnen uiteraard ook weggelaten worden. Indien externe RAM wordt aangesloten gebruikt de controller P0 als gemultiplexte adres/data-bus. Tijdens een geheugenactie (lezen of schrijven) wordt eerst de low-byte van het adres met behulp van het ALE-signaal in een latch gezet die op P1 is aangesloten. Daarna wordt de data gelezen of geschreven. De high-byte van het adres wordt via P2 uitgevoerd.
Poort P0 is toegankelijk via connector J2.

connector P1

pen	bit	functie
1		GND (aarde = 0 Volt)
2		VCC (voedingsspanning = 5 Volt)
3	P1.0	i/o, segment A
4	P1.1	i/o, segment B
5	P1.2	i/o, segment C
6	P1.3	i/o, segment D
7	P1.4	i/o, segment E
8	P1.5	i/o, segment F
9	P1.6	i/o, segment G
10	P1.7	i/o, segment DP



In de practicumcomputer is poort P1 verbonden met een 7-segmentdisplay en tevens met een connector met de naam P1. Door een "1" op een pen te zetten gaat het bijbehorende segment uit. Een "0" daarentegen laat het segment branden. Het logische niveau op de pennen van de poort, en daarmee op de connector en het 7-segmentdisplay, kan dus zowel door het programma van de computer als door een randapparaat worden bepaald.
Poort P1 is toegankelijk via connector J3.

connector P2

pen	bit	functie
1		GND (aarde = 0 Volt)
2		VCC (voedingsspanning = 5 Volt)
3	P2.0	i/o, A8
4	P2.1	i/o, A9
5	P2.2	i/o, A10
6	P2.3	i/o, A11
7	P2.4	i/o, A12
8	P2.5	i/o, A13
9	P2.6	i/o, A14
10	P2.7	i/o, A15

Poort P2 is een bidirectionele datapoort met interne pull-up weerstanden. Indien externe RAM wordt aangesloten gebruikt de controller P0 als gemultiplexte adres/data-bus en P2 om de high-byte van het adres uit te voeren. Tijdens een geheugenactie (lezen of schrijven) wordt eerst de low-byte van het adres met behulp van het ALE-signaal via P0 in een latch gezet terwijl de high-byte van het adres in P2 wordt gezet. Daarna wordt de data gelezen of geschreven.
Poort P2 is toegankelijk via connector J4.

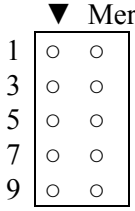
connector P3

pen	bit	functie
1		GND (aarde = 0 Volt)
2		VCC (voedingsspanning = 5 Volt)
3	P3.0	i/o, RxD (serial input), via jumper J7
4	P3.1	i/o, TxD (serial output), via jumper J8
5	P3.2	i/o, INT0# (interrupt input), druktoets TEST
6	P3.3	i/o, INT1# (interrupt input)
7	P3.4	i/o, T0 (timer 0 input)
8	P3.5	i/o, T1 (timer 1 input)
9	P3.6	i/o, WR# extern RAM
10	P3.7	i/o, RD# extern RAM

betekent 'actief laag', d.w.z. er is geen actie als het signaal hoog is.

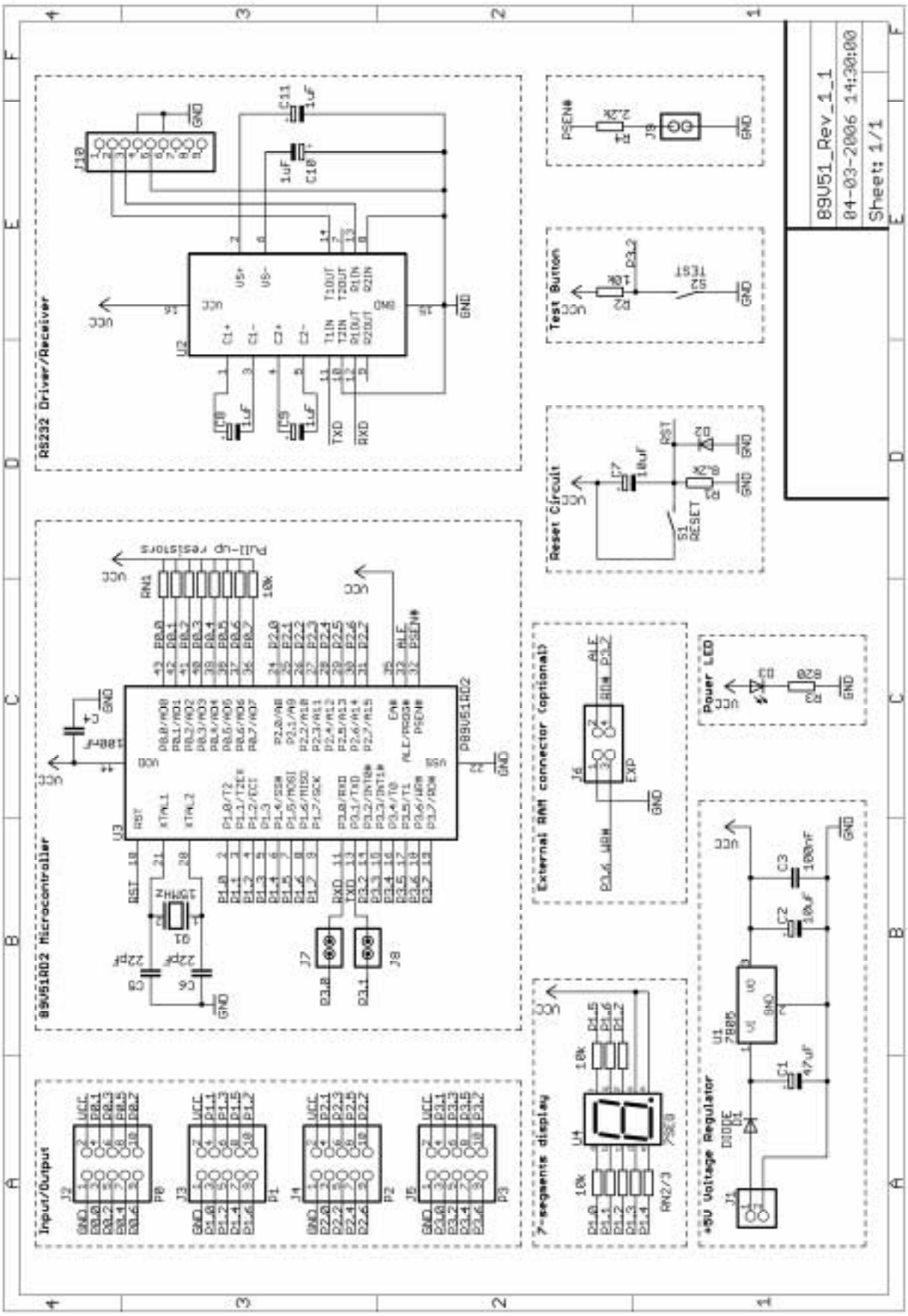
De poortbits P3.0 t/m P3.7 kunnen als algemene input/output bit gebruikt worden maar ze hebben ook andere functies. Bijvoorbeeld P3.0 en P3.1 zijn bestemd voor seriële communicatie. De signalen P3.0 en P3.1 kunnen met behulp van de jumpers P7 en P8 worden doorgegeven aan de connectorpennen 3 en 4 (zie hierboven). Op P3.2 is een drukschakelaar met de naam TEST aangesloten. Door de schakelaar in te drukken wordt een "0" aangeboden. In rust geeft de schakelaar een "1".
Poort P3 is toegankelijk via connector J5.

Het voorgaande overzicht geeft de aansluitmogelijkheden van de computer weer. De pennummers van de poortconnectors zijn te vinden in de volgende figuur. Het is gebruikelijk dat pen 1 wordt gemarkeerd door een merkteken, bijvoorbeeld een pijltje of een stip.



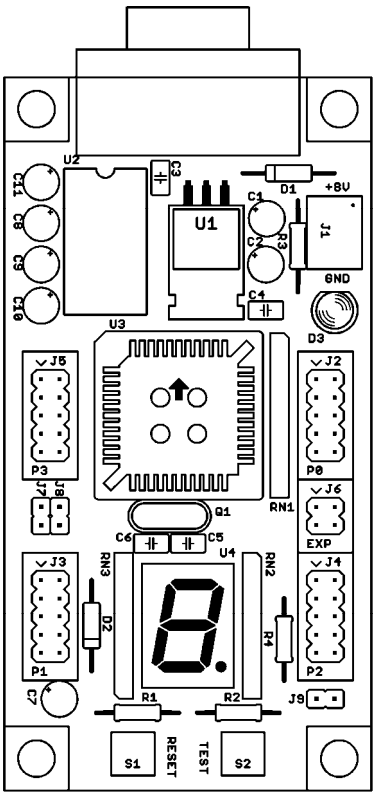
Pennummers van de connectors

1.4 Het schema van de 89V51 computer



1.5 De componentenopstelling van de 89V51 computer

De afbeelding hieronder laat de componentenopstelling zien van het controllerbordje. De codes bij de componenten verwijzen naar het schema in paragraaf 1.4 en naar de onderdelenlijst in paragraaf 1.6.



1.6 Onderdelen en kostprijs van de 89V51 computer

Component	Fabrikant	Beschrijving	Leverancier	Ordercode	RoHS	Aantal	Prijs [(1)]	PCB code
RE3-47V470M	ELNA	CAPACITOR, 47µF 35V	FARNELL	3618602	N	1	0,048/1000	C1
RE3-16V100M	ELNA	CAPACITOR, 10µF 16V	FARNELL	3618420	N	2	0,043/1000	C2, C7
RE3-100V010M	ELNA	CAPACITOR, 1µF 100V	FARNELL	3618857	N	4	0,042/1000	C8, C9, C10, C11
MCDR25104X7RK0050	MULTICOMP	CAPACITOR, 100nF 50V	FARNELL	750920		2	0,06/1000	C3, C4
MCDR25220COGJ0200	MULTICOMP	CAPACITOR, 22pF 200V	FARNELL	746990		2	0,08/1000	C5, C6
1N4007	ON SEMICONDUCTOR	DIODE, 1A 1000V	FARNELL	3525340	N	1	0,03/1000	D1
1N914	FAIRCHILD SEMICONDUCTOR	DIODE, SMALL SIGNAL DO-35	FARNELL	885060		1	0,02/1000	D2
HLMP-4700	AGILENT TECHNOLOGIES	LED, 5mm RED	FARNELL	323135	N	1	0,143/1000	D3
SFR25 8K2 5%	VISHAY BC COMPONENTS	RESISTOR, 0.4W 5% 8K2	FARNELL	332800	N	1	0,03/300	R1
SFR25 10K 5%	PHOENIX PASSIVE COMPONENTS	RESISTOR, 0.4W 5% 10K	FARNELL	9476300	J	1	0,03/300	R2
SFR25 820R 5%	VISHAY BC COMPONENTS	RESISTOR, 0.4W 5% 820	FARNELL	332689		1	0,02/300	R3
SFR25 2K2 5%	VISHAY BC COMPONENTS	RESISTOR, 0.4W 5% 2K2	FARNELL	332732		1	0,02/300	R4
4609X-101-103LF	BOURNS	RESISTOR NETWORK, 10K	FARNELL	9356819	J	1	0,114/1000	RN1
4608X-102-222	BOURNS	RESISTOR NETWORK, 2K2	FARNELL	107059	N	2	0,114/1000	RN2, RN3
LF A158K	C-MAC	CRYSTAL HC49/4H 12.0000 MHz	FARNELL	9712950	J	1	0,47/250	Q1
MC7805CT	ON SEMICONDUCTOR	IC, REGULATOR +5.0V	FARNELL	701853	N	1	0,24/250	U1
ST232CN	STMICROELECTRONICS	IC, TRANSCEIVER	FARNELL	3025020	N	1	0,57/500	U2
1-390261-4	TYCO ELECTRONICS	DIL SOCKET 16 P	FARNELL	1101347	J	1	0,08/1000	(U2)

P89V51RD2	PHILIPS [2]	IC, 8-BIT FLASH MCU	Philips			1	10,00 [2]	U3
HDSP-5551	AGILENT TECHNOLOGIES	LED DISPLAY, 0.56" HE RED	FARNELL	324759	N	1	0,54/1000	U4
M20-9970546	HARWIN	HEADER EXTENDED 2 ROW 5W	FARNELL	621869	J	4	0,22/500	J2,J3,J4,J5
8LCM009S-304B-XX	MULTICOMP	SOCKET, D PCB R/A 9WAY	FARNELL	4106118		1	0,34/500	J10
		PLCC SOCKET 44P	CONRAD	189731		1	0,77/100	(U3)
		MINI PULSTOETS KORTE TOETS	CONRAD	700460		1	0,31/50	S1
		MINI PULSTOETS LANGE TOETS	CONRAD	700479		1	0,31/50	S2
		AANSLUITKLEM 2-POLIG	CONRAD	729949		1	0,33/100	J1
		CYLINDERBOUT 16mm, M3	CONRAD	815586		4	0,015/100	
		ZESKANTMOER M3	CONRAD	815624		4	0,009/100	
	CYNER SUBSTRATES	PRINTED CIRCUIT BOARD	CYNER		N	1	1,75/1000	
	TU DELFT EWI	TOTAAL ONDERDELEN					8,60 [3]	
		SOLDEER, REPARATIE			N		1,40	
	TU DELFT EWI	KOSTPRIJS PRACTICUM					**10,00 [3]**	

Leveranciers

Farnell	Farnell InOne	(www.farnell.com)
Conrad	Conrad Nederland	(www.conrad.nl)
Cyner	Cyner Substrates	(www.cyner.nl)

Voetnoten

1) Prijzen zijn als volgt genoteerd: staffelprijs/staffelaantal. Prijzen Farnell: ex BTW, Conrad en Cyner: incl BTW.
2) De microcontroller is voor het practicum gratis ter beschikking gesteld door Philips.
3) Zonder microcontroller, inclusief BTW.

1.7 Schemabeschrijving

De processor
Het hart van de schakeling is de 89V51RD2 microcontroller U3, met als klok een kristaloscillator waartoe het 12 MHz kristal Q1 en de twee condensatoren C5 en C6 behoren.

Adres- en databus
Er is geen systeembus voor adressen en data omdat er geen extern geheugen aanwezig is.

Programma- en datageheugen
Omdat pen EA# met VCC is verbonden haalt de processor instructies uit het interne programmageheugen als de waarde van de Program Counter ligt binnen het adresgebied van dit geheugen. Daarbuiten worden instructies met behulp van het signaal PSEN# (het teken # betekent dat het signaal actief laag is) uit een eventueel aanwezig extern programmageheugen gehaald. De 89V51RD2 heeft 64K intern programmageheugen waardoor extern geheugen geen zin heeft. Bij versies met minder intern programmageheugen kan extra programmageheugen worden aangesloten. De 89V51RD2 heeft 768 bytes RAM dat als extern datageheugen moet worden geadresseerd.

Het 7-segmentdisplay
Datapoort P1 stuurt een 7-segment display. De zeven segmenten en de decimale punt zijn lichtgevende diodes (LEDs) die al goed zichtbaar zijn bij een stroom van 1,6 mA. De segmenten hebben een gemeenschappelijke anodeaansluiting die met VCC is verbonden. In serie met ieder segment is een weerstand opgenomen die de grootte van de stroom door de LED bepaalt. Deze weerstanden zitten per vier in de SIL (Single In Line) netwerken RN2 en RN3.

Datapoort P1 en connector J3
Op poort P1 is behalve het display ook connector J3 aangesloten. Op deze connector zijn tevens de 5 Volt voedingsspanning (VCC) en aarde (GND) beschikbaar. Een op connector P1 aangesloten schakeling kan de benodigde voeding uit de computer betrekken. De datapoort is bidirectioneel, dus voor zowel input als output geschikt.

Datapoort P3 en connector J5
Op connector J5 zijn tevens de 5 Volt voedingsspanning (VCC) en aarde (GND) beschikbaar. Een op connector P1 aangesloten schakeling kan de benodigde voeding uit de computer betrekken. De datapoort is bidirectioneel, dus voor zowel input als output geschikt. De pennen 3 en 4 van de connector kunnen met de jumpers J7 en J8 worden verbonden met de signalen P3.0 en P3.1. Deze signalen worden door de controller gebruikt bij seriële communicatie o.a. bij het flashen van een programma. Voorzichtigheid is geboden als externe apparatuur wordt aangesloten op J5.

Datapoort P0 en connector J2
Op connector J2 zijn tevens de 5 Volt voedingsspanning (VCC) en aarde (GND) beschikbaar. Een op connector P0 aangesloten schakeling kan de benodigde voeding uit de computer betrekken. De datapoort is bidirectioneel, dus voor zowel input als output geschikt. De pull-up weerstanden zijn extern aangebracht maar kunnen desgewenst worden weggelaten.

Datapoort P2 en connector J4
Op connector J4 zijn tevens de 5 Volt voedingsspanning (VCC) en aarde (GND) beschikbaar. Een op connector P2 aangesloten schakeling kan de benodigde voeding uit de computer betrekken. De datapoort is bidirectioneel, dus voor zowel input als output geschikt.

Reset-circuit
De processor krijgt een reset-signaal uit een circuit bestaande uit drukschakelaar RESET, weerstand R1, en condensator C7 en diode D2. Ook bij het inschakelen van de voedingsspanning wordt een resetpuls gegenereerd (Power-On-Reset).

Drukschakelaar TEST
Bit 2 van poort P3 is aangesloten op drukschakelaar TEST. Met de schakelaar in rust staat er een "1" op P3.2, ingedrukt staat er een "0" op P3.2. De schakelaar is niet ontdenderd. Men kan de schakelaar gebruiken als input-orgaan om een eenvoudige applicatie te bedienen.

De voeding
Het voedingsdeel van de computer bestaat uit de 5 Volt spanningsstabilisator U1. De uitgangsspanning van deze regelaar is 5 Volt (VCC) waarmee de gehele computer wordt gevoed. De ingangsspanning van U1 moet groter zijn dan 7,5 Volt. Diode D1 beschermt de computer tegen verkeerd aansluiten van de voedingsspanning. De condensatoren C1, C2 en C3 zorgen voor de afvlakking van rimpels op de voeding en de stabiliteit van de voedingsspanning.

De power LED
Er is een LED aangebracht om te kunnen zien of de voedingsspanning aanwezig is. De stroom door LED D3 wordt bepaald door weerstand R3.

RS232 interface

U2 is een seriële "line driver/line receiver" die tussen de UART in de controller en de seriële externe dataverbinding is geplaatst. Via de 9-polige connector J10 kan de UART in de controller worden geprogrammeerd. Een gebruikersprogramma kan via de UART met een aangesloten serieel apparaat communiceren, bijvoorbeeld een terminal. De interface werkt als inverter en als line-driver/line-receiver. De controller werkt met signalen op 5 Volt-niveau. De signaalniveaus op de communicatieverbinding zijn conform de RS232 standaard: logisch 1 is -9 Volt en logisch 0 is +9 Volt. De transmitter is aangesloten op pen 2 van de connector. De receiver op pen 3. Pen 5 is aarde. De vier condensatoren C8, C9, C10 en C11 horen bij de gelijkspannings-converters in de RS232-interface waarmee de benodigde interne spanningen van 10 Volt en -10 Volt worden gemaakt waarmee het mogelijk wordt de positieve en negatieve spanningen van het uitgangssignaal te maken.

Flash-circuit voor een 89C51Rx2
In plaats van een 89V51RD2 kan een 89C51Rx2 worden gemonteerd. Het circuit met R4 en J9 is nodig om deze controller in de flash-mode te zetten.

1.8 Soldeervoorschriften

Goed solderen is een kunst die moet worden geoefend. Voor het solderen gebruiken we soldeertin met een harskern. Het hars laat het soldeer goed vloeien. Zorg dat de punt van de bout goed schoon is (gebruik het vochtige sponsje). Neem het soldeertin in de linkerhand en de soldeerbout in de rechter (of omgekeerd). Houd de punt van de bout gedurende niet meer dan twee seconden tegen de te solderen aansluiting. Duw gedurende niet langer dan een seconde het soldeertin tegen de aansluiting en de bout. Het soldeertin moet goed vloeien. Haal daarna eerst het soldeertin en kort daarna de bout van de verbinding. Even wachten en/of een beetje blazen.

Adem de rook van de verdampte hars niet in!

1.9 Montage

Bekijk eerst alle onderdelen en probeer ze te herkennen. Kijk goed naar de positie van ieder onderdeel alvorens het definitief vast te solderen. De bovenkant van de print (componentenzijde) bevat een witte opdruk die de positie van de componenten laat zien.
Begin de montage met de lage onderdelen en daarna de hogere. Dit maakt het mogelijk om de print plat op tafel te leggen bij het monteren van ICs en voeten. Gebruik zonodig de printklem om het wegglijden van het werkstuk te voorkomen.
Buig lange draden aan de onderkant van de print een beetje om zodat het onderdeel er niet uit kan vallen. Soldeer daarna het onderdeel vast en knip de draden af met een kniptang.
De aanbevolen volgorde van het monteren van de onderdelen is als volgt:

<u>De weerstanden R1, R2, R3 en R4</u>
Raadpleeg de kleurcodetabel in paragraaf 1.10 voor het herkennen van de weerstanden. Buig de aansluitdraden op de goede afstand 90 graden om met een platte tang en steek de component in de print. Buig daarna de draden 45 graden zodat de component niet meer uit de print kan vallen.
R1 is 8.2 kohm
R2 is 10 kohm
R3 is 820 ohm
R4 is 2.2 kohm (ontbreekt op de foto in WinUptools)

<u>De diodes D1 en D2</u>
Let bij de diodes op de polariteit. De grijze of zwarte ring geeft de kathode aan. Op de print is de kathode met een dwars streepje aangegeven. D1 heeft een zwarte behuizing. D2 is kleiner en heeft een roodachtige behuizing.

<u>Het kristal Q1</u>
Bij een hoog model: zorg ervoor dat de opdruk van het kristal aan de kant van de condensatoren komt zodat het zichtbaar blijft. Het lage model met de opdruk op de bovenkant mag niet te lang verhit worden!

<u>De condensatoren C5 en C6</u>
Deze kleine condensatoren met een waarde van 22 pF maken deel uit van de kristaloscillator. Ze hebben geen polariteit. Ze zijn kleiner dan C3 en C4 waardoor de opdruk te klein is om met het blote oog te kunnen lezen.

<u>De spanningsregulator U1</u>
Deze component heeft een metalen koelvlak waarin een montagegat zit. De pennen vlak bij de verdikking 90 graden ombuigen en wel zo dat de opdruk naar boven zichtbaar blijft.

De weerstandsnetwerken RN1, RN2 en RN3
Dit zijn de gele componenten met 8 of 9 aansluitingen op een rij. RN1 heeft een oriëntatie. De zwarte punt komt bij het schuine kantje in de opdruk op de print. Deze component bevat acht weerstanden van 10 kohm die aan een kant met elkaar zijn verbonden. Dit is de aansluiting bij de zwarte punt. Deze component heeft 9 pennen.
RN2 en RN3 hebben geen oriëntatie, hoewel de zwarte punt anders doet vermoeden. Deze component heeft 8 pennen en bevat vier weerstanden van 2.2 kohm die niet met elkaar zijn verbonden.

De condensatoren C3 en C4
Deze componenten hebben geen polariteit. Deze hoogfrequent ontkoppelcondensatoren hebben een waarde van 100nF. Ze zijn iets groter dan C5 en C6.

De 16 pens DIL-voet voor U2
De inkeping op een van de korte zijden is aangegeven in de opdruk op de print. Het IC (MAX232 of ST232CN) wordt later geplaatst als het hele bordje klaar is.

De 44-pens PLCC-voet voor U3
LET OP! Het schuine vlakje op een van de hoeken is ook in de opdruk op de print terug te vinden.
De microcontroller wordt later geplaatst als het hele bordje klaar is.

Het 7-segment display U4
Let op de oriëntatie. De decimale punt is op de print aangegeven.

De LED D3
Deze component heeft een polariteit. Het rechte vlakje is op de print is aangegeven. De plus-aansluiting heeft een langere draad.

Het aansluitblokje J1
Deze component heeft twee schroefjes waarmee de voedingsspanning kan worden aangesloten. Let er op dat de gaatjes in de zijkant aan de rand van de print komen.

De elektrolitische condensatoren C1, C2, C7, C8, C9, C10 en C11
Deze componenten hebben een polariteit. De langere draad is de plus-aansluiting. Op de print is de plus aangegeven met het teken +. De min-draad en de waarde van de component zijn op de component aangegeven.
C1 is 47 µF
C2 en C7 zijn 10µF
C8, C9, C10 en C11 zijn 1µF

De druktoetsen S1 en S2
Druktoets RESET heeft een korte toets en druktoets TEST heeft een lange toets.

De connectors J2, J3, J4, en J5
Deze connectors bestaan uit twee rijen met vijf pennen. Ze worden met de korte kant van de pennen in de print gesoldeerd.

Connector J10
De 9-polige haakse D-sub connector wordt als laatste component op de print gemonteerd. De montage-pennen passen in twee gaten met een soldeerbare rand. Zet ze met een beetje soldeer vast. Probeer niet de gaten geheel met soldeer te vullen.

Niet monteren
De componenten J6, J7, J8 en J9 bevinden zich niet in het standaard bouwpakket.

Plaats de microcontroller
Leg deze component zorgvuldig in de IC-voet. De enigszins afgeschuinde hoek (goed kijken) past bij de schuine kant aan de binnenkant van de IC-voet. Probeer zo weinig mogelijk met de vingers aan de contactvlakjes te komen. Druk nu met twee duimen de controller in de voet. Ondersteun de print daarbij, bijvoorbeeld met de wijsvingers.

Plaats de MAX232
Buig de twee rijen pennen van deze component haaks door ze tegelijk op de tafel om te buigen. Niet met de vingers aankomen. Druk het IC in de voet waarbij de uitholling op de korte zijde overeenkomt met de uitholling in de voet.

1.10 Kleurcode-tabel voor weerstanden

kleur	*cijfer*	*vermenigvuldiger*	*tolerantie*
	ring 1 en 2	**ring 3**	**ring 4**
zwart	0	1 x	
bruin	1	10 x	1%
rood	2	100 x	2%
oranje	3	1.000 x	
geel	4	10.000 x	
groen	5	100.000 x	
blauw	6	1.000.000 x	
violet	7	-	
grijs	8	-	
wit	9	-	
zilver	-	0,01 x	10%
goud	-	0,1 x	5%

Voorbeeld: geel/violet/rood/goud = 4700 ohm 5%

Weerstanden worden gemaakt in een aantal standaardreeksen. De E12 reeks bevat 12 verschillende standaard weerstandswaarden per decade, aangegeven door de eerste twee kleurcoderingen. De waarden zijn: 10 12 15 18 22 27 33 39 47 56 68 82.
Er komen ook weerstanden voor met een kleurring extra. De eerste drie kleurringen geven dan de standaardwaarde aan. De vierde kleurring geeft de vermenigvuldigfactor en de vijfde de tolerantie.

1.11 Aansluitgegevens

Als de spanningsstabilisator niet extra wordt gekoeld mag de voedingsspanning niet hoger zijn dan 25 Volt . Als op de stabilisator een koelplaatje wordt gemonteerd mag de voedingsspanning wat hoger zijn, maar nooit hoger dan 35 Volt. De temperatuurverhoging van het inwendige van de stabilisator mag de 65 graden Celcius niet overschrijden. De opgenomen voedingsstroom van de computer is 35 mA als de I/O-pennen niet zijn belast en ook het 7-segment display uit is.

De computer kan op eenvoudige wijze worden gevoed uit een netspanningsadapter die een gelijkspanning produceert van 8 Volt of hoger en die minimaal 2 maal de opgenomen stroom kan leveren. Zo wordt de adapter niet maximaal belast. De connector van een dergelijke adapter moet worden afgeknipt, de draden gestript en vervolgens vastgezet in het aansluitblokje van de computer. Let daarbij op de polariteit. Voor kort durende experimenten kan een 9-Volt alkaline-batterij worden gebruikt.

Attentie:
Een wisselspanningsadapter (AC-AC adapter) of een beltrafo is niet zonder meer geschikt als voedingsspanningsapparaat.

1.12 Het testen van de computerhardware

Na het bouwen van de computer moet de goede werking worden vastgesteld.

Sluit de voeding aan. De LED moet gaan branden. Druk op de RESET knop.

Er zijn vier gaatjes te zien tussen J2 en J4 (P0 en P2). Dit is connector J6 (EXP). Controleer met een oscilloscoop het ALE signaal dat op aansluiting 2 van connector J6 gemeten kan worden. Dit is het gaatje rechtsboven. Er moet een blokgolf te zien zijn met een frequentie van 12/6 MHz = 2 MHz.

Het 7-segmentdisplay moet geheel gedoofd zijn.

Roep uw assistent erbij als er iets niet goed is.

1.13 Functionele test met een testprogramma

Er is een testprogramma voorhanden waarmee een aantal onderdelen van de computer kunnen worden getest. Dit programma heet 89V51TEST. De hex-file 89V51Test.hex staat in de werkmap op de PC. In paragraaf 1.14 staat de procedure die moet worden gevolgd om een programma in het flash-geheugen van de microcontroller te zetten.

Flash nu het testprogramma.
Na het flashen moet op de druktoets RESET worden gedrukt waardoor het testprogramma door de controller wordt uitgevoerd. Als alles goed is, gaan alle segmenten nu een voor een aan en uit.

Door op de druktoets TEST te drukken kunnen verschillende hardware functies van de computer worden getest. Het volgordecijfer van de test wordt op het 7-segment display getoond. De volgende lijst geeft aan wat er wordt getest.

cijfer	functie
1	test druktoets, de 1 verschijnt niet als de toets niet werkt
2	pulssignaal met pulsbreedte van 1 msec op alle pennen van poort van P0, P2 en P3
3	geïnverteerde 1 msec puls op alle pennen van poort P0, P2 en P3

Door op TEST te drukken kan tussen test 2 en test 3 heen en weer worden geschakeld.

Roep uw assistent erbij als er iets niet klopt.

1.14 Flash procedure

Verbind de seriële kabel met de 9-polige connector van het microcontrollerbord.
Ga via Start/Programma's naar IO Elektronica en start het flash-programma FlashMagic.
Controleer de instellingen in de vakken 1 t/m 4 en wijzig deze zo nodig.

In vak 1:	COM Port	COM 1
	Baudrate	4800
	Device	89V51RD2
In vak 2:	kies Erase blocks used by Hex File	
	merk op dat de 80V51RD2 slechts één blok heeft	
In vak 3:	selecteer de file 89V51Test.hex	
In vak 4:	geen enkele optie kiezen	
In vak 5:	start het flashen	

Als alles goed is geeft FlashMagic nu de aanwijzing 'Reset the device into ISP mode now'.
Druk nu op de knop RESET van het microcontrollerbord.
NB: Als de microcontroller al in de flash-mode staat verschijnt deze boodschap niet en start het flashen automatisch.

Het wissen van de gebruikte geheugenblokken en het flashen van het programma vinden achtereenvolgens plaats. Let op het statusbericht linksonder. Als het flashen is beëindigd wordt de melding 'Finished' getoond.

Pas op:
Als FlashMagic vraagt:
"You have selected to program the clock bit which once programmed cannot be erased using ISP, a parallel programmer is required. Are you sure you want to program the clock bit ?"
Antwoord:
"No"

1.15 Een programma executeren

Druk na het flashen opnieuw op RESET om het geflashte programma te starten. De microcontroller begint na een reset altijd met de instructie op adres 0.

1.16 Programma-ontwikkeling

Er zijn verschillen methoden om de machinecode waaruit een programma bestaat te genereren, te weten: handmatig coderen, assembleren en compileren. Op het practicum wordt de eerstgenoemde methode gebruikt bij het uitvoeren van de eerste programmeeropdracht. Daarna wordt de assembler gebruikt.

1.17 Handmatig coderen

Kleine programma's kunnen worden gemaakt zonder een assembler te gebruiken. Hierbij moeten we zelf de machine-code in de vorm van hexadecimale getallen bepalen en opschrijven. Vervolgens wordt een (Flash)EPROM met deze gegevens geprogrammeerd.

Allereerst worden de instructies in assembly-taal opgeschreven waarbij van labels gebruik gemaakt mag worden. Daarna wordt van iedere instructie de machinecodebytes bepaald en genoteerd. De lijst met instructies (de instructie set) in deze handleiding is daarbij onmisbaar.

Om het coderen overzichtelijk te houden wordt een geschikt formulier gebruikt. Achter in deze handleiding bevindt zich een aantal van deze formulieren. Het eerste formulier is voorzien van een voorbeeldprogramma.

Het coderen met behulp van dit formulier verloopt als volgt:

- Noteer van iedere benodigde instructie de kolommen LABEL, OPCODE en OPERAND.
- Bepaal daarna van boven naar beneden het machinecodebyte dat de opcode representeert. Noteer dit byte in kolom BYTE 1 in hexadecimale vorm.
- Stel vast uit hoeveel bytes een instructie bestaat en gebruik dit gegeven om de kolom ADRES in te vullen, te beginnen met het adres waar het programma moet beginnen.
- Als de instructie een operand bevat moet die ook vertaald worden. De operand komt in BYTE 2 en eventueel in BYTE 3 te staan.

Achterwaartse sprongen kunnen worden opgelost als de kolom ADRES netjes is ingevuld, immers bij ieder label van een vertaalde instructie is nu het adres bekend. Omdat een voorwaartse adresreferentie nog niet kan worden opgelost laten we de operand-bytes nog even open. Nadat het gehele programma is vertaald kunnen de opengelaten sprongadressen worden ingevuld.

De kolommen BYTE 1, BYTE 2 en BYTE 3 bevatten de machinecode van het programma. Deze bytes worden op de PC ingetypt en daarna in een hex-file opgeslagen. Vervolgens wordt de hex-file in de microcontroller geflashed.

Voor het handmatig programmeren van de microcontroller wordt het volgende recept gebruikt:

- Kies op de PC Start/Programma's/IO Elektronica/WinUptools2 (versie 2).
- Druk op de knop Hex-File.
- Stel de Target Zone in van 0000 t/m 001F.
- Vul de machinecode-bytes in waaruit het programma bestaat.
- Druk op de knop Save TZ en geef een filenaam op met extensie .hex, bijvoorbeeld Opdracht1.hex. Hiermee wordt een hex-file gemaakt met daarin de ingevoerde machinecode.
- Flash het programma met FlashMagic. Zie hiervoor H1.14.

1.18 Assembleren

Eerst wordt met behulp van een editor een tekstfile gemaakt waarin de instructies in assembly-taal worden genoteerd. Met een vertaalprogramma, genaamd assembler, wordt deze tekstfile omgezet in een hex-file. Deze hex-file bevat de machinecode-bytes waaruit het programma bestaat, geordend in records die voorzien zijn van de adressen waarop de machinecode-bytes in het programmageheugen zijn gealloceerd. In H3.9 wordt uitgelegd hoe een hex-file is opgebouwd.
Met een flash-programma worden de machinecode-bytes vervolgens in het flash-geheugen van de microcontroller gezet.

De assembler wordt in het volgende hoofdstuk behandeld. Een effectief gebruik van de assembler wordt bevorderd door het betreffende hoofdstuk grondig te bestuderen. Er is niet overwogen om de oorspronkelijke documentatie die in het Engels is gesteld, te vertalen omdat in de computerwereld deze taal veel wordt gebruikt. Het is leerzaam daar vanaf het begin aan te wennen.

Het volgende recept geeft aan welke stappen moeten worden gezet om een programma te ontwikkelen en in een (Flash)EPROM te zetten. In dit recept wordt het programma WinUptools gebruikt waarmee een programma kan worden geschreven en geassembleerd.

Recept

1. Op de PC: Start/Programma's/IO Elektronica/WinUptools2.
   Druk op de knop Source File in het hoofdmenu van WinUptools.
   Voer een bestandsnaam in met het commando, bij voorkeur met de extensie .A51, bijvoorbeeld OPDR2.A51

2. Start de editor met het commando Edit. De default editor is Notepad. Creëer een nieuwe file.

3. Tik het programma in. Gebruik alleen hoofdletters en gebruik de TAB-toets om rechte kolommen te maken. Neem boven in dit tekstbestand een ORG directief op dat verwijst naar de locatie in de (Flash)EPROM waar het programma geplaatst moet worden. Dit adres is 0000H voor een lege (Flash)EPROM.
   Voorbeeld: ORG 0200H

4. Bewaar de source-file en sluit de editor af. De file OPDR2.A51 wordt op de harde schijf bewaard.

5. Assembleer de file OPDR2.A51 door het commando Assemble te geven. De assembler genereert een list-file (OPDR2.LST), een binaire file (OPDR2.BIN) en een hex-file (OPDR2.HEX).

6. Als de assembler fouten meldt, moeten die eerst worden opgelost alvorens verder te gaan.
   - Open de editor, verbeter de fouten en sluit de editor weer af..
   - Assembleer de source-file opnieuw.

7. Bekijk de list-file OPDR2.LST met het commando View.
   Lijkt alles te kloppen? Ook het beginadres?
   Bekijk ook de hex-file OPDR2.HEX.

8. Gebruik het programma FlashMagic om de hex-file in de microcontroller te programmeren.
   Op de PC: Start/Programma's/IO Elektronica/FlashMagic. Zie ook H1.14.

2 Assembler Documentation

2.1 Introduction

The ASM51EP assembler is a program development tool used to translate a 8051 assembly language source file into an object code file containing the corresponding machine code. Two different types of object code files can be generated. The first type is binary coded and suitable to be programmed in an EPROM. This EPROM has to be placed in the 8051 'target' computer. The second type is a so called 'hex' file holding machine code information in readable ASCII text. This file could be transported (down loaded) to a target computer where the machine code portion is converted to the actual binary machine code and placed in RAM by a running program (absolute loader).

In short: the assembler inputs an assembly source file and outputs a list file, a hex file, and a binary file. Option parameters are available to choose from two types of hex formats and two types of binary formats. This will be explained later.

The source file is translated during two passes. While reading the file for the first time the assembler stores encountered labels and their corresponding addresses in a 'symbol table' and translates instructions into machine code. During the second pass references to labels are resolved and output files are generated.

The assembler keeps track of its most important task - where to put what in program memory - by means of a pointer, called the 'location counter'. After each translation of an instruction the location counter marks the location (memory address) of the next byte to be stored in program memory. Note that the assembler's location counter and the processor's program counter have completely different functions.

Apart from storing machine code in memory the assembler can be instructed to allocate and fill memory with data. This data is also placed in program memory. If the computer system has separate program and data memory areas, data placed in program memory can be read only, not written. However, if the memory areas are overlapping there is no physical difference in the actual storage location. Consequently data stored in program memory can be accessed (read and written) as if it had been stored in data memory.

2.2 Assembly line format

An assembly source-line generally consists of three parts:

```
Label          Instruction     Comment

example:
Lab_1:         MOV             A,#12          ;load accumulator with number
```

If the first character in a line is one of the letters A to Z, or a to z , the first symbol of the line is taken as a label. Labels may be up to 9 characters long, and may contain the underscore symbol (_). If the first character is a space or a tab character, the assembler decides that there is no label. Labels must be delimited by a space, a tab or a colon (:). Note that a delimiting colon is not a part of the label and is not counted either.

The assembler is case sensitive with respect to labels (i.e. upper case and lower case characters are recognized as different characters). The -U option is used when the assembler has to treat upper case and lower case characters in labels equally. See the paragraph on command line options for more

details.The instruction part of the line has to be in upper case. It is interpreted as an assembler pseudo-instruction or machine-language instruction.
When a semicolon (;) appears anywhere in the line, the text following it is interpreted as comment.

2.3 Arithmetic expressions

Numerical constants and Character constants are arithmetic expressions.
Also labels are arithmetic expressions.
$ is a special arithmetic expression, its value is the value of the location counter.

The assembler does not recognize parenthesis '()'.
The integer range is 0 to 65535.

When expr1 and expr2 are arithmetic expressions, then also:

```
-expr1          ; a negative value
expr1 + expr2   ; Sum
expr1 - expr2   ; Difference
expr1 * expr2   ; Product
expr1 / expr2   ; Integer Division (quotient part of divide result)
expr1 % expr2   ; Modulo Division (remainder part of divide result)
```

Examples:
3+7*2 equals 3+(7*2) and gives 17.
17/3 gives 5
17%3 gives 2

2.4 Notation of numerical and alphabetic constants

Numerical constants	:			
Decimal numbers	:	normal notation.		
Hexadecimal numbers	:	marked by suffix H, must start with a number 0 to 9		
Binary numbers	:	marked by suffix B.		
Character constants	:	a character enclosed by single quotes.		
String constants	:	a string of characters enclosed by single quotes.		
Examples	:			
Decimal numbers	:	0	00001	12324
Hexadecimal numbers	:	0CH	0A3H	8ABCH
Binary numbers	:	11B	011B	00000011B
Character constants	:	'A'	'0'	'!'
String constants	:	'XYZ'	'1234H'	'this is text',0

2.5 Pseudo-Instructions

The following pseudo-instructions are assembler directives. They control the assembler but do not generate (executable) machine code.

```
            ORG         expr
            DB          expr,expr,expr ...
            DW          expr,expr,expr ...
            DS          expr
Label       EQU         expr
            END
```

Short description
The expression 'expr' is evaluated to a number.
ORG initializes (or sets) the location counter the value of 'expr'.
DB inserts bytes in a program and advances the location counter accordingly.
DW inserts double bytes in a program and advances the location counter accordingly.
DS only advances the location counter with the value of 'expr'. No data is stored.
EQU assigns the value of 'expr' to a symbol.

ORG (Origin)
The instruction

```
            ORG         expr
```

assigns the value of the arithmetic expression to the location counter. This instruction is used to make the program start or continue at a specific location in the memory. In the next example ORG is used to define the start address of a program at location 100H.

Example:

```
            ORG         100H
START       LJMP        SOMEWHERE
```

DB (Define Byte)
The instruction

```
            DB          expr,expr,....
```

saves the values of the arithmetic expressions as bytes in the program memory, and increments the location counter as required. String constants are also allowed. The next example shows how bytes and string values may be combined.

Example:

```
COMMAND     DB          13,10,'COMMAND',0  ;CR,LF,String,Delimiter
```

This line places the (hex) byte values 0D,0A,43,4F,4D,4D,41,4E,44,00 in program memory in a consecutive order. The zero-byte marks the end of the string.

DW (Define Word)
The instruction

```
            DW          expr,expr,....
```

saves the values of the arithmetic expressions as two byte values in the program memory, and increments the location counter as required. The byte order is high byte at the current location counter followed by the low byte. String constants are also allowed. The following example shows the use of this directive.

Example:

```
            DW          1234H,ABCH,'ABC'
```

resulting in the consecutive byte values 12,34,0A,BC,41,42,43 (hex) in program memory.

DS (Define Storage)

The instruction

```
        DS      expr
```

increments the location counter with the value of the arithmetic expression. If the program is to be placed in external RAM and program memory an data memory are physically identical the defined storage area may be used to store variables. In the next example DS is used to define a data storage area in external RAM at location 1000H.

Example:

```
        ORG     1000H
NUMBER  DS      2       ; reserve 2 bytes for variable NUMBER
RESULT  DS      4       ; reserve 4 bytes for variable RESULT
KEY     DS      1       ; reserve 1 byte for variable KEY
```

To read or write a variable in external data memory use the MOVX instruction:

```
        MOV     DPTR,#RESULT    ; initialize Data Pointer first
        MOVX    A,@DPTR         ; read first byte of RESULT
        MOVX    @DPTR,A         ; write first byte of RESULT
                                ; (DPTR)=1002H
```

Directive DS may also be used, in combination with ORG instructions, to manage the internal RAM memory. This only works because DS does not leave any data in the binary and hex output files. The assembler assigns values to the labels which can be used in the program. In the next example DS is used to define the same data storage area in internal RAM at location 50H.

Example:

```
        ORG     50H
NUMBER  DS      2       ; reserve 2 bytes for variable NUMBER
RESULT  DS      4       ; reserve 4 bytes for variable RESULT
KEY     DS      1       ; reserve 1 byte for variable KEY
```

To read or write a variable in internal RAM use the MOV instruction with direct addressing.

```
        MOV     A,RESULT    ; read first byte of RESULT
                            ; address of RESULT is 52H
```

Note that the same method is used in both cases. However, variables have to be accessed differently.

EQU (Equate)

The instruction

```
LABEL   EQU     expr
```

assigns the value of the arithmetic expression to identifier LABEL. In the next example is shown how to use it to define constants.

Example:

```
MIN     EQU     100
MAX     EQU     1000+MIN-1  ; evaluates a simple expression
```

An alternative method to organize variables space in internal RAM uses the EQU directive in stead of the DS directive. In the next example EQU is used to define the addresses in internal RAM of several variables.

Example:

```
POINTER     EQU         50H     ; reserve 2 bytes for variable POINTER
RESULT      EQU         52H     ; reserve 4 bytes for variable RESULT
KEY         EQU         56H     ; reserve 1 byte for variable KEY
```

The same method can be used to define space for variables in external data memory. Again, the MOVX instruction has to be used to access variables in external RAM.

END

The instruction

```
            END
```

marks the end of the source file. The assembler discards all lines following the END directive.

2.6 Bit addressing

The bit addressable locations may be specified with their actual bit addresses.

Example:

```
            CLR         21H     ; clear bit 1 of location 20H
            CPL         0E7H    ; complement bit 7 of the accumulator
            JB          95H     ; jump if port P1 bit 5 is logical '1'
```

An alternative way uses the assemblers bit specifier '.' following the byte address in which the bit is located.

Example:

```
ACC         EQU         0E0H    ; address of accumulator
P1          EQU         090H    ; address of port P1
SW5         EQU         P1.5    ; bit address of port P1 bit 5
            CLR         20H.1   ; clear bit 1 of location 20H
            CPL         ACC.7   ; complement bit 7 of the accumulator
            JB          P1.5    ; jump if port P1 bit 5 is logical '1'
            JB          SW5     ; jump if port P1 bit 5 is logical '1'
```

The last line in this example shows the use of a fully symbolic bit address to specify a bit operand. This method has clearly many advantages over the blunt way of bit addressing used in the previous example. Be careful not to use instuctions for bit addressing when accessing byte locations. The results will be unpredictable.

Example:

```
ACC         EQU         0E0H    ; address of accumulator
            CLR         ACC     ; does not clear the whole accumulator
            CLR         A       ; this clears the accumulator
```

2.7 List file

The listing produced by the assembler has the following structure:

```
****** LISTING of ASM51EP (C:ADDN) ******
LINE   LOC    OBJ     T       SOURCE
  1    0000                   ; SUBROUTINE ADDN
  2    0000                   ; FUNCTIE:          Optellen van getallen met lengte N
  3    0000                   ;                   a + b -> a
  4    0000                   ; INPUT             R0 = pointer naar low byte van a
  5    0000                   ;                   R1 = pointer naar low byte van b
  6    0000                   ;                   R7 = aantal bytes per getal (is N)
  7    0000                   ; OUTPUT:           C = Carry
  8    0000                   ;                   R0 = pointer naar low byte van som
  9    0000                   ; VERANDERT:        A,C,R1,R7
 10    0000                   ;
 11    0000   C0 00   [2]     ADDN    PUSH  0     ; bewaar R0 op de stack
 12    0002   C3      [1]             CLR   C     ; zet de carry op 0
 13    0003   E6      [1]     ADD1    MOV   A,@R0 ; haal byte van getal a op
 14    0004   37      [1]             ADDC  A,@R1 ; tel byte van b op + Carry
 15    0005   F6      [1]             MOV   @R0,A ; zet som terug
 16    0006   08      [1]             INC   R0    ; verhoog pointer naar a
 17    0007   09      [1]             INC   R1    ; verhoog pointer naar b
 18    0008   DF F9   [2]             DJNZ  R7,ADD1 ; verlaag aantal, klaar?
 19    000A   D0 00   [2]             POP   0     ; herstel R0
 20    000C   22      [2]             RET         ; keer terug naar aanroeper
 21    000D
 22    000D                           END
```

The column "LINE" contains the line number of the source text, in decimal notation.
The column "LOC" (Location) indicates the value of the location (address) counter (4 digit hexadecimal).
The column "OBJ" indicates the generated object code in hexadecimal notation.
The column "T" shows the execution time of instructions in cycles. At a clock of 12 MHz, one cycle lasts exactly one microsecond.
The column "SOURCE" shows the assembly-language source code.

2.8 HEX file

The assembler generates a readable object-code file that forms a "HEX DUMP". Each line of this file has a specific format according to the selected type of hex file. You may select the well known universal INTEL hex format or the assembler specific ASM51EP hex format. How this is done will be explained later.

INTEL hex format

Each line in an INTEL hex file consists of the following items:

:NATBB.....BC<CR><LF>

Where:

:	Start of line character.
N	Number of object-code bytes to follow (as a 2-digit hexadecimal number).
A	Address of first data byte to follow (as a 4-digit hexadecimal number).
T	Type of line (as a 2-digit hexadecimal number). 00 is a line containing object-code bytes 01 is the last line (no object-code bytes) 03 is a line holding the programs start address
B	These are the object-code bytes, each as a two-digit hexadecimal number.
C	Check-sum. 2's complement value of the modulo 256 sum of all preceding bytes with the exception of ':' (as a 2-digit hexadecimal number).

The CR (Carriage Return) and LF (Line Feed) characters are not required. However they give the receiver time to process the line before the next line comes in.

Example (spaces added here to show the field boundaries):
:10 0800 00 310029010000CD0100010200CD030001 EB
:00 0000 01 FF

ASM51EP type hex format

Each line in a hex file consists of the following items, separated by spaces and ending with a CR and a LF character:

N A:B B . . . . . B C<CR><LF>

Where:

N	Number of object-code bytes to follow (as a 2-digit hexadecimal number).
A	Address of bytes to follow (as a 4-digit hexadecimal number).
:	Separator character.
B	These are object-code bytes, each as a two-digit hexadecimal number, separated by spaces.
C	Sum of object-code bytes as a 4-digit hexadecimal number (check-sum).

Example:
07 1000:E8 22 31 32 33 34 00 01D4<CR><LF>

2.9 Binary file

The assembler generates (writes) a binary file that forms an "Object Code Dump". The file structure and length is specific for the selected type of binary file.

Standard binary file:
Only one ORG directive at the beginning of the source file is allowed using this type of binary output. Object-code bytes are written to the binary file in a consecutive order. Absolute load addresses are not included. The DS directive does not put any data in this type of binary file.

Binary memory image file:
More than one ORG directive may be used in the source file. Object-code bytes are written to the binary file in such a way that an image of an 8 Kbyte (EP)ROM is created. Fill bytes (with value FF) are written from address 0 to the first code byte, between program blocks and after the last code byte to the end of the 8 Kbyte memory space. The DS directive inserts fill bytes too. The relative location of each byte in the file is equal to its absolute address modulo 2000H.

2.10 Error reports of ASM51EP

1	string length	Indicated text string is too long.
2	missing or invalid term	Missing valid arithmetic expression.
3	unknown expression	Arithmetic expression can not be computed.
4	missing expression	An arithmetic expression must follow.
5	missing term	Missing arithmetic term.
6	to many bytes	Line generates too many opcode bytes.
7	range error	Arithmetic expression does not fit in BYTE
8	syntax	Syntax error.
8	unknown address	Indicated address (or label) is not known.
10	relative jump range	Jump out of range.
11	far jump range	Jump out of range.
12	base address not bit addressable	as is.
13	unknown opcode	as is.
14	invalid expression	as is.
15	missing label	as is.
16	double defined label	as is.
17	invalid line	Line ending not allowed.
18	division by zero	Zero divide in expressions.
19	invalid bit number	Wrong bit number in bit addressing.
20	invalid register	Wrong register in register indirect addressing.
21	end of file encountered	No END directive found.

2.11 Voorbeeldprogramma

De volgende listing toont het programma 89V51TEST.A51 dat wordt gebruikt om de computer te testen. Het is zeer aan te bevelen om dit programma goed te bestuderen.

```
LINE LOC  OBJ       T  SOURCE
   1 0000              ;POORTADRESSEN
   2 0000 00 80        P0      EQU     080H
   3 0000 00 90        P1      EQU     090H
   4 0000 00 A0        P2      EQU     0A0H
   5 0000 00 B0        P3      EQU     0B0H
   6 0000
   7 0000              ;DEFENITIE VAN DE DRUKTOETS
   8 0000 B2           TEST    EQU     P3.2
   9 0000
  10 0000              ;HULPVARIABELE
  11 0000 00 30        TEMP    EQU     030H
  12 0000
  13 0000              ;STARTADRES
  14 0000                      ORG     0000H
  15 0000
  16 0000              ;ROTEER SEGMENT
  17 0000 74 FE    [1] TEST0   MOV     A,#11111110B    ;11111110 ALLEEN BIT 0 AAN
  18 0002 F5 90    [1] LOOP    MOV     P1,A            ;STUUR DIT NAAR HET DISPLAY
  19 0004 23       [1]         RL      A               ;ROTEER BITS NAAR LINKS
  20 0005 12 00 70 [2]         LCALL   WAIT300         ;WACHT 200 MS
  21 0008 20 B2 F7 [2]         JB      TEST,LOOP       ;TEST OF TOETS IS INGEDRUKT
  22 000B 12 00 5B [2]         LCALL   WAIT50          ;ONTDENDER DE TOETS
  23 000E
  24 000E              ;TEST TOETS
  25 000E 30 B2 FD [2] TEST1   JNB     TEST,$          ;WACHT OP LOSLATEN TOETS
  26 0011 12 00 5B [2]         LCALL   WAIT50          ;ONTDENDER DE TOETS
  27 0014 75 90 F9 [2]         MOV     P1,#11111001B   ;ZET CIJFER ÉÉN OP DISPLAY
  28 0017 20 B2 FD [2]         JB      TEST,$          ;WACHT OP INDRUKKEN TOETS
  29 001A 12 00 5B [2]         LCALL   WAIT50          ;ONTDENDER
  30 001D 30 B2 FD [2] TEST23  JNB     TEST, $         ;WACHT OP LOSLATEN TOETS
  31 0020 12 00 5B [2]         LCALL   WAIT50          ;ONTDENDER
  32 0023
  33 0023              ;BITTEST VAN DE POORTEN MET 1 BIT HOOG
  34 0023              ;ELKE MS WORDT EEN VOLGEND BIT VAN EEN POORT HOOG GEMAAKT
  35 0023              ;DIT RESULTEERT IN EEN SIGNAAL MET EEN DUTY-CYCLE VAN 1/8
  36 0023 75 90 A4 [2] TEST2   MOV     P1,#10100100B   ;ZET CIJFER TWEE OP DISPLAY
  37 0026 74 01    [1]         MOV     A,#001H         ;MAAKT EEN PIN HOOG
  38 0028 F5 80    [1] LOOP2   MOV     P0,A            ;STUUR TESTPATROON NAAR P0
  39 002A F5 A0    [1]         MOV     P2,A            ;EN NAAR POORT P2
  40 002C F5 30    [1]         MOV     TEMP,A
  41 002E 44 07    [1]         ORL     A,#00000111B    ;MASKEER P3.0, P3.1 EN p3.2
  42 0030 F5 B0    [1]         MOV     P3,A            ;EN NAAR POORT P3
  43 0032 E5 30    [1]         MOV     A,TEMP
  44 0034 23       [1]         RL      A               ;ROTEER BITS
  45 0035 12 00 68 [2]         LCALL   WAIT1           ;WACHT 1 MS
  46 0038 20 B2 ED [2]         JB      TEST,LOOP2      ;ALS TOETS NIET INGEDRUKT
  47 003B 12 00 5B [2]         LCALL   WAIT50          ;ONTDENDER
  48 003E 30 B2 FD [2]         JNB     TEST,$          ;WACHT OP LOSLATEN TOETS
  49 0041
  50 0041              ;BITTEST VAN DE POORTEN MET 1 BIT LAAG
  51 0041              ;ELKE MS WORDT EEN VOLGEND BIT VAN EEN POORT LAAG GEMAAKT
  52 0041              ;DIT RESULTEERT IN EEN SIGNAAL MET EEN DUTY-CYCLE VAN 7/8
  53 0041 75 90 B0 [2] TEST3   MOV     P1,#10110000B   ;HET CIJFER DRIE OP DISPLAY
  54 0044 74 FE    [1]         MOV     A,#0FEH         ;MAAK EEN BIT LAAG
  55 0046 F5 80    [1] LOOP3   MOV     P0,A
  56 0048 F5 A0    [1]         MOV     P2,A
  57 004A F5 30    [1]         MOV     TEMP,A
  58 004C 44 07    [1]         ORL     A,#00000111B
  59 004E F5 B0    [1]         MOV     P3,A
  60 0050 E5 30    [1]         MOV     A,TEMP
  61 0052 23       [1]         RL      A
  62 0053 12 00 68 [2]         LCALL   WAIT1
  63 0056 20 B2 ED [2]         JB      TEST,LOOP3
  64 0059 80 C2    [2]         SJMP    TEST23          ;HERHAAL BITTEST TOT RESET
  65 005B
```

```
66 005B                ;SUBROUTINE WAIT50, EXECUTIETIJD=50
67 005B 78 82    [1] WAIT50  MOV     R0,#130         ;130X128X3=50000 MICROSEC
68 005D 79 80    [1] LUS2    MOV     R1,#128
69 005F 19       [1] LUS1    DEC     R1              ;1 MICROSEC
70 0060 B9 00 FC [2]         CJNE    R1,#00,LUS1     ;2 MICROSEC
71 0063 18       [1]         DEC     R0
72 0064 B8 00 F6 [2]         CJNE    R0,#00,LUS2
73 0067 22       [2]         RET
74 0068
75 0068                ;SUBROUTINE WAIT1, EXECUTIETIJD=1
76 0068 78 FA    [1] WAIT1   MOV     R0,#250         ;250X4=1000 MICROSEC
77 006A 00       [1] LUS     NOP                     ;1 MICROSEC
78 006B 18       [1]         DEC     R0              ;1 MICROSEC
79 006C B8 00 FB [2]         CJNE    R0,#00,LUS      ;2 MICROSEC
80 006F 22       [2]         RET
81 0070
82 0070                ;SUBROUTINE WAIT200, EXECUTIETIJD=300
83 0070 7A 06    [1] WAIT300 MOV     R2,#6
84 0072 12 00 5B [2] W3      LCALL   WAIT50
85 0075 DA FB    [2]         DJNZ    R2,W3
86 0077 22       [2]         RET
87 0078
88 0078                      END
```

3 Machine codes

In de volgende tabellen wordt een overzicht gegeven van de 8051 instructies. In de kolom INSTRUCTION staat de assembly-notatie. In de kolom OPERATION staat wat er gebeurt als de instructie wordt uitgevoerd. In kolom PSW worden de (mogelijke) veranderingen in het Processor Status Word vermeld. Kolom T geeft het aantal machinecycli en kolom L het aantal bytes waaruit de instructie bestaat. Tenslotte geeft kolom CODE de opcode in binaire vorm (dit is het eerste byte van de instructie). In de tabellen worden de volgende symbolen gebruikt:

Symbool	Betekenis
#	immediate adressering
data	8 bit waarde
data16	16 bit waarde
Rn	R0,R1,R2,R3,R4,R5,R6 of R7 van de huidige geselecteerde registerbank geheugenlocatie in bank 0 heeft R0 adres 0, R1 adres 1 etc. in bank 1 heeft R0 adres 8, R1 adres 9 etc. etc.
rrr	3 bit waarde 000,001,010,011,100,101,110 of 111
Ri	R0 of R1 van de huidige geselecteerde registerbank
@Ri	indirecte adressering van een interne geheugenlocatie via R0 of R1 0-127 bij een 128 byte intern datageheugen 0-255 bij een 256 byte intern datageheugen
direct	8 bit intern adres geheugenlocatie (0-127) of SFR (128-255)
bit	direct geadresseerd bit in intern datageheugen of in een SFR in datageheugen: bitadressen 0-127 bit=0 betekent geheugenadres 20H bit 0 bit=127 betekent geheugenadres 2FH bit 7 bit=25H.1 betekent geheugenadres 25H bit 1 in een SFR: binaire adressen eindigend op 000 (80H, 88H, 90H etc.) bit=87H betekent SFR adres 80H bit 7 bit=80H.7 betekent SFR adres 80H bit 7
rel	2's complement 8 bit offset byte relatief t.o.v. het eerste byte van de volgende instructie rel=FEH betekent 2 bytes lager rel=2 betekent 2 bytes hoger
addr11	11 bit adres aaa = A10,A9,A8 A7-A0 in tweede byte

addr16	16 bit adres in tweede (high byte) en derde byte (low byte)
A_n	bit n van A, n = 0-7
A_{0-3}	bit 0 t/m 3 van A
not	logische NOT (inversie of 1's complement) bit: not 0 = 1, not 1 = 0 byte: not 11110000 = 00001111
and	logische AND bit: 0 and X = 0, 1 and 1 = 1 byte: 11110000 and 00110011 = 00110000
or	logische OR bit: 1 or X = 1, 0 or 0 = 0 byte: 11110000 or 11001100 = 11111100
exor	logische EXCLUSIVE OR bit: 0 exor 1 = 1, 0 exor 0 = 0, 1 exor 1 = 0 carry from A6 exor carry from A7 levert "1" op bij ongelijkheid borrow to A6 exor borrow to A7 levert "1" op bij ongelijkheid byte: 11110000 exor 11001100 = 00111100
(direct)	de inhoud van de geheugenplaats met het adres 'direct'
(Ri)	de inhoud van de geheugenplaats met het adres dat in Ri staat.
←	toekenning A←Rn betekent: de inhoud van Rn wordt gekopieerd in A (SP) ← (direct) betekent: de inhoud van adres 'direct' wordt gekopieerd in de geheugenplaats waar de stackpointer naar wijst. C←0 betekent: C-vlag wordt altijd 0
↔	verwisseling A↔Rn betekent: de inhouden van Rn en A worden verwisseld

DATA TRANSFER					
INSTRUCTION	OPERATION	PSW	T	L	OPCODE
MOV A,Rn	A←Rn	-	1	1	11101rrr
MOV A,direct	A←(direct)	-	1	2	11100101
MOV A,@Ri	A←(Ri)	-	1	1	1110011i
MOV A,#data	A←data	-	1	2	01110100
MOV Rn,A	Rn←A	-	1	1	11111rrr
MOV Rn,direct	Rn←(direct)	-	2	2	10101rrr
MOV Rn,#data	Rn←data	-	1	2	01111rrr
MOV direct,A	(direct)←A	-	1	2	11110101
MOV direct,Rn	(direct)←Rn	-	2	2	10001rrr
MOV direct,direct	(direct)←(direct)	-	2	3	10000101
MOV direct,@Ri	(direct)←(Ri)	-	2	2	1000011i
MOV direct,#data	(direct)←data	-	2	3	01110101
MOV @Ri,A	(Ri)←A	-	1	1	1111011i
MOV @Ri,direct	(Ri)←(direct)	-	2	2	1010011i
MOV @Ri,#data	(Ri)←data	-	1	2	0111011i
MOV DPTR,#data16	DPTR←data16	-	2	3	10010000
MOVC A,@A+DPTR	A←(A+DPTR)	-	2	1	10010011
MOVC A,@A+PC	A←(A+PC)	-	2	1	10000011
MOVX A,@Ri	A←(Ri)	-	2	1	1110001i
MOVX A,@DPTR	A←(DPTR)	-	2	1	11100000
MOVX @Ri,A	(Ri)←A	-	2	1	1111001i
MOVX @DPTR,A	(DPTR)←A	-	2	1	11110000
PUSH direct	SP←SP+1 (SP)←(direct)	-	2	2	11000000
POP direct	(direct)←(SP) SP←SP-1	-	2	2	11010000
XCH A,Rn	A↔Rn	-	1	1	11001rrr
XCH A,direct	A↔(direct)	-	1	2	11000101
XCH A,@Ri	A↔(Ri)	-	1	1	1100011i
XCHD A,@Ri	A_{0-3}↔(Ri_{0-3})	-	1	1	1101011i

ARITHMETIC OPERATIONS					
INSTRUCTION	OPERATION	PSW	T	L	OPCODE
ADD A,Rn	A←A+Rn	C←carry from A7 AC←carry from A3 OV←carry from A6 exor carry from A7	1	1	00101rrr
ADD A,direct	A←A+(direct)	C AC OV	1	2	00100101
ADD A,@Ri	A←A+(Ri)	C AC OV	1	1	0010011i
ADD A,#data	A←A+data	C AC OV	1	2	00100100
ADDC A,Rn	A←A+Rn+C	C AC OV	1	1	00111rrr
ADDC A,direct	A←A+(direct)+C	C AC OV	1	2	00110101
ADDC A,@Ri	A←A+(Ri)+C	C AC OV	1	1	0011011i
ADDC A,#data	A←A+data+C	C AC OV	1	2	00110100
SUBB A,Rn	A←A-Rn-C	C←borrow for A7 AC←borrow for A3 OV←borrow for A6 exor borrow for A7	1	1	10011rrr
SUBB A,direct	A←A-(direct)-C	C AC OV	1	2	10010101
SUBB A,@Ri	A←A-(Ri)-C	C AC OV	1	1	1001011i
SUBB A,#data	A←A-data-C	C AC OV	1	2	10010100
INC A	A←A+1	-	1	1	00000100
INC Rn	Rn←Rn+1	-	1	1	00001rrr
INC direct	(direct)←(direct)+1	-	1	2	00000101
INC @Ri	(Ri)←(Ri)+1	-	1	1	0000011i
INC DPTR	DPTR←DPTR+1	-	2	1	10100011
DEC A	A←A-1	-	1	1	00010100
DEC Rn	Rn←Rn-1	-	1	1	00011rrr
DEC direct	(direct)←(direct)-1	-	1	2	00010101
DEC @Ri	(Ri)←(Ri)-1	-	1	1	0001011i
MUL AB	unsigned multiply A←low byte of A*B B←high byte of A*B	C←0 OV←1 if A*B>255 OV←0 if A*B<=255	4	1	10100100
DIV AB	unsigned divide A←integer part of A/B B←remainder of A/B	C←0 OV←1 if divide by 0 OV←0 if not	4	1	10000100
DA A	A←decimal adjust A following ADD or ADDC in packed BCD format	C←1 if result>99 C←0 if result<=99 OV←0	1	1	11010100

BOOLEAN VARIABLE MANIPULATION					
INSTRUCTION	OPERATION	PSW	T	L	OPCODE
CLR C	C←0	C←0	1	1	11000011
CLR bit	(bit)←0	-	1	2	11000010
SETB C	C←1	C←1	1	1	11010011
SETB bit	(bit)←1	-	1	2	11010010
CPL C	C←not C	C←not C	1	1	10110011
CPL bit	(bit)←not(bit)	-	1	2	10110010
ANL C,bit	C←C and (bit)	C	2	2	10000010
ANL C,/bit	C←C and not(bit)	C	2	2	10110000
ORL C,bit	C←C or (bit)	C	2	2	01110010
ORL C,/bit	C←C or not(bit)	C	2	2	10100000
MOV C,bit	C←(bit)	C	1	2	10100010
MOV bit,C	(bit)←C	-	2	2	10010010
JC rel	PC←PC+2 if C=1 then PC←PC+rel	-	2	2	01000000
JNC rel	PC←PC+2 if C=0 then PC←PC+rel	-	2	2	01010000
JB bit,rel	PC←PC+3 if (bit)=1 then PC←PC+rel	-	2	3	00100000
JNB bit,rel	PC←PC+3 if (bit)=0 then PC←PC+rel	-	2	3	00110000
JBC bit,rel	PC←PC+3 if (bit)=1 then PC←PC+rel and (bit)←0	-	2	3	00010000

PROGRAM BRANCHING					
INSTRUCTION	OPERATION	PSW	T	L	OPCODE
ACALL addr11	PC←PC+2 SP←SP+1 (SP)←PC_{7-0} SP←SP+1 (SP)←PC_{15-8} PC_{10-0}←addr11	-	2	2	aaa10001
LCALL addr16	PC←PC+3 SP←SP+1 (SP)←PC7-0 SP←SP+1 (SP)←PC15-8 PC_{15-0}←addr16	-	2	3	00010010
RET	PC_{15-8}←(SP) SP←SP-1 PC_{7-0}←(SP) SP←SP-1	-	2	1	00100010
RETI	PC_{15-8}←(SP) SP←SP-1 PC_{7-0}←(SP) SP←SP-1	-	2	1	00110010
AJMP addr11	PC←PC+2 PC_{10-0}←addr11	-	2	2	aaa00001
LJMP addr16	PC_{15-0}←addr16	-	2	3	00000010
SJMP rel	PC←PC+2 PC←PC+rel	-	2	2	10000000
JMP @A+DPTR	PC←A+DPTR	-	2	1	01110011
JZ rel	PC←PC+2 if A=0 then PC←PC+rel	-	2	2	01100000
JNZ rel	PC←PC+2 if A<>0 then PC←PC+rel	-	2	2	01110000
CJNE A,direct,rel	PC←PC+3 if A<>(direct) then PC←PC+rel if A<(direct) then C←1 else C←0	C	2	3	10110101
CJNE A,#data,rel	PC←PC+3 if A<>data then PC←PC+rel if A<data then C←1 else C←0	C	2	3	10110100
CJNE Rn,#data,rel	PC←PC+3 if Rn<>data then PC←PC+rel if Rn<data then C←1 else C←0	C	2	3	10111rrr

CJNE @Ri,#data,rel	PC←PC+3
	if (Ri)<>data then PC←PC+rel
	if (Ri)<data then C←1 else C←0
		C	2	3	1011011i

DJNZ Rn,rel	PC←PC+2 Rn←Rn-1 if Rn<>0 then PC←PC+rel	-	2	2	11011rrr
DJNZ direct,rel	PC←PC+3 (direct)←(direct)-1 if (direct)<>0 then PC←PC+rel	-	2	3	11010101

LOGICAL OPERATIONS FOR BYTE VARIABLES					
INSTRUCTION	OPERATION	PSW	T	L	OPCODE
ANL A,Rn	A←A and Rn	-	1	1	01011rrr
ANL A,direct	A←A and (direct)	-	1	2	01010101
ANL A,@Ri	A←A and (Ri)	-	1	1	0101011i
ANL A,#data	A←A and data	-	1	2	01010100
ANL direct,A	(direct)←(direct) and A	-	1	2	01010010
ANL direct,#data	(direct)←(direct) and data	-	2	3	01010011
ORL A,Rn	A←A or Rn	-	1	1	01001rrr
ORL A,direct	A←A or (direct)	-	1	2	01000101
ORL A,@Ri	A←A or (Ri)	-	1	1	0100011i
ORL A,#data	A←A or data	-	1	2	01000100
ORL direct,A	(direct)←(direct) or A	-	1	2	01000010
ORL direct,#data	(direct)←(direct) or data	-	2	3	01000011
XRL A,Rn	A←A exor Rn	-	1	1	01101rrr
XRL A,direct	A←A exor (direct)	-	1	2	01100101
XRL A,@Ri	A←A exor (Ri)	-	1	1	0110011i
XRL A,#data	A←A exor data	-	1	2	01100100
XRL direct,A	(direct)←(direct) exor A	-	1	2	01100010
XRL direct,#data	(direct)←(direct) exor data	-	2	3	01100011
CLR A	A←0	-	1	1	11100100
CPL A	A_n←not A_n (n=0-7)	-	1	1	11110100
RL A	A_{n+1}←A_n (n=0-6) A_0←A_7	-	1	1	00100011
RLC A	A_{n+1}←A_n (n=0-6) A_0←C C←A_7	-	1	1	00110011
RR A	A_n←A_{n+1} (n=0-6) A_7←A_0	-	1	1	00000011
RRC A	A_n←A_{n+1} (n=0-6) A_7←C C←A_0	-	1	1	00010011
SWAP A	A_{3-0}↔A_{7-4}	-	1	1	11000100
NOP	No operation	-	1	1	00000000

4 Notatie van getallen

De schrijfwijze van een cijfer of een getal hangt af van het grondtal. Een decimaal getal heeft grondtal 10, daaraan zijn we gewend. Een hexadecimaal getal heeft grondtal 16 (decimaal geschreven) en een binair getal heeft grondtal 2. De binaire en de hexadecimale schrijfwijze komen in de computerwereld veel voor. De binaire komt overeen met de manier waarop getallen in een computer worden opgeslagen en verwerkt, namelijk als reeksen bits die de waarde 0 of 1 kunnen hebben. De hexadecimale schrijfwijze is een compacte notatiewijze voor deze onoverzichtelijke bitpatronen.

In de tabel hieronder zijn voor getallen van 0 tot 15 de verschillende notaties naast elkaar gezet.

decimaal	**hexadecimaal**	**binair**
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

De volgende tabel laat een beperkt aantal grotere getallen zien.

decimaal	**hexadecimaal**	**binair**
16	10	10000
32	20	100000
64	40	1000000
128	80	10000000
255	FF	11111111
256	100	100000000
512	200	1000000000
1024	400	10000000000
2048	800	100000000000
4096	1000	1000000000000
8192	2000	10000000000000
16384	4000	100000000000000
32768	8000	1000000000000000
65535	FFFF	1111111111111111
65536	10000	10000000000000000

Getallen zonder en met teken (unsigned en signed)

Bij getallen zonder teken (unsigned binary) maakt ieder cijfer van dat getal deel uit van de getalswaarde.

Bijvoorbeeld:
234 dec = $2 \times 10^2 + 3 \times 10^1 + 4 = 234$
A3C hex = $10 \times 16^2 + 3 \times 16^1 + 12 = 2620$ dec
101 bin = $1 \times 2^2 + 0 \times 2^1 + 1 = 5$

In getallen met een teken wordt het hoogstwaardige bit gebruikt om het teken aan te geven. Er bestaan verschillende binaire coderingen, onder meer de 2's complement codering en de excess codering (ook wel offset binary genoemd).

De volgende tabel laat deze twee coderingen zien voor 8 bit getallen. Bit 7 is het tekenbit.

decimaal	**hexadecimaal**	**2's complement**	**excess 128**
127	7F	01111111	11111111
126	7E	01111110	11111110
//			
3	03	00000011	10000011
2	02	00000010	10000010
1	01	00000001	10000001
0	00	00000000	10000000
-1	FF	11111111	01111111
-2	FE	11111110	01111110
-3	FD	11111101	01111101
//			
-127	81	10000001	00000001
-128	80	10000000	00000000

De excess code lijkt op de unsigned binary code maar de getalswaarde is 2^{n-1} lager (n is het aantal bits, inclusief het tekenbit). Hier komt de naam 'offset binary' vandaan. Bij een 8 bit getal wordt de code 'excess 128' genoemd. De excess code komt onder meer voor bij Digitaal-Analoog convertors. De laagste waarde komt overeen met de meest negatieve analoge spanning die de convertor kan genereren (bijv. -5 Volt), de hoogste waarde komt overeen met de meest positieve spanning (bijv. +5 Volt). De excess code kan eenvoudig in de 2's complement code worden omgezet (en vice versa) door het tekenbit te inverteren.

De 2's complement code wordt vaak gebruikt als er gerekend moet worden. De 8051 processor is zo ontworpen dat de vlaggen in het Processor Status Word betrekking hebben op bewerkingen van zowel unsigned als 2's complement getallen. Afhankelijk van het gebruikte getalformaat moet de bijbehorende vlag getest worden. Voor unsigned getallen is dat de C-vlag en voor signed getallen de OV-vlag.

Voorbeeld:
na 80+7F is C=0 en OV=0 (geen unsigned en geen signed overflow)
na FF+01 is C=1 en OV=0 (unsigned overflow maar geen signed overflow)
na 7F+01 is C=0 en OV=1 (geen unsigned overflow maar signed overflow)
na 80+80 is C=1 en OV=1 (unsigned overflow en signed overflow)

Naam: Do Cent
Studienummer: vergeten
Datum: 29 aug 2002

Programma: Rollend Segment

File: ROLSEG.BIN

ADRES	BYTE 1	BYTE 2	BYTE 3	LABEL	OPCODE	OPERAND
0000	74	FE			MOV	A,#11111110B
0002	23			START1	RL	A
0003	F5	90			MOV	P1,A
0005	12	00	0D		LCALL	WACHT
0008	20	B2	F7		JB	P3.2,START1
000B	80	FE		EINDE	SJMP	EINDE
000D	78	FF		WACHT	MOV	R0,#0FFH
000F	79	FF		BUITLUS	MOV	R1,#0FFH
0011	19			BINLUS	DEC	R1
0012	B9	00	FC		CJNE	R1,#00H,BINLUS
0015	18				DEC	R0
0016	B8	00	F6		CJNE	R0,#00H,BUITLUS
0019	22				RET	

Naam: Programma:
Studienummer:
Datum: File:

Naam: Programma:
Studienummer:
Datum: File:
