MSX turbo R: de processor DE R800 ONTSLUIERD

MSX Computer Magazine nummer 43 - december 1990

Scanned, ocr’ed and converted to PDF by HansO, 2001

In MSX Computer Magazine nummer 41 schreven we het al: er bestaat een opvolger voor de MSX2+. De turbo R standaard doet zijn intrede. Ditmaal geen grote veranderingen op grafisch gebied. Wat er verbeterd is, is de verwerkingssnelheid: de turbo R bevat een nieuwe processor.

Sinds ML-redacteur Markus The kennis heeft aan Didi Hirokawa - inderdaad, een Japanse naam - hebben de Japanse bladen en documentatie weinig geheimen meer voor ons. Handig, zo'n relatie. Vooral als we de hand weten te leggen op de specificaties van die geheimzinnige R800 processor, het hart van de nieuwe turbo R.

Opvolger

De R800-processor is nog het beste te beschouwen als een soort supersnelle Z80. Door allerlei oorzaken is de R800 een stuk sneller dan de Z80, bovendien kent deze processor een aantal nieuwe instructies. Het meest opvallende is het snelheidsverschil. De R800 is beduidend sneller en krachtiger dan de Z80, dat staat buiten kijf. De R800 is opcode-compatibel met de Z80, de R800 kan alle machinetaal-instructies van de Z80 uitvoeren. Voor een gemiddeld programma is de R800 echter wel zo'n vier tot vijf keer sneller dan de Z80.
Dat zal overigens ook de reden zijn dat er nog steeds een Z80 is de turbo R zit. In principe zouden alle bestaande MSX programma's ook door de R800 uitgevoerd moeten kunnen worden, maar wanneer de snelheid nauw luistert kunnen er toch wat vreemde effecten optreden. Om dat op te lossen kan er altijd teruggeschakeld worden naar de oude trouwe Z80.
Maar meestal zal in zo'n turboR machine de R800 actief zijn. Dat chipje is immers een stuk sneller en biedt meer mogelijkheden. Maar waar komt dat hoge tempo vandaan?

Alle beetjes helpen

Ten eerste is de kloksnelheid van de R800 hoger. De Z80 in een MSX draait op 3.57 MegaHertz, terwijl de R800 op 7.16 MHz werkt. Maar intern werkt de nieuwe processor met een klokfrequentie die nog eens vier keer zo hoog is, zodat hij feitelijk op 28.64 MHz draait.
Ten tweede is de R800 een 16 bits processor, althans intern. De interne databus is 16 bits, de externe is nog steeds 8 bits breed. Daardoor kunnen bijvoorbeeld de MSX sloten gewoon in gebruik blijven en zijn er geen (dure) 16 bits support chips nodig. In feite is de R800 hetzelfde opgebouwd als de Intel 8088, de chip waar de eerste IBM PC's rond zijn opgebouwd. Het betekent dat 16 bits woorden - registerparen en

adressen -binnen de chip dubbel zo snel kunnen worden getransporteerd.
Ten derde gaat de R800 beter met het geheugen om. Wanneer het hoge byte van het geheugenadres tussen twee aanroepen van het geheugen niet verandert, is de R800 twee keer zo snel dan normaal. Het hoge byte wordt niet opnieuw op de adresbus geplaatst. Dit wordt 'page mode RAM access' genoemd. Tenslotte kan de R800 nog een heel moderne snelheids-truuk gebruiken, die we de laatste tijd ook steeds meer in de duurdere PC's zien toegepast. Het is namelijk een feit, dat RAM sneller is dan ROM. De R800 kan hier gebruik van maken door de BIOS-, Basic-, SUB- en Kanji-ROM naar RAM te kopiëren. Dit stuk RAM wordt dan beschermd tegen schrijfopdrachten, zodat het er voor de R800 uitziet als ROM. Dit wordt de 'R800 DR AM-mode' genoemd.

Interne snelheid

Op grond van de kloksnelheid zou de R800 acht keer sneller moeten zijn dan de Z80. Maar dat gaat natuurlijk slechts ten dele op. Sommige instructies zullen zelfs meer dan acht keer zo snel zijn, omdat 16 bits gegevens binnen de chip sneller getransporteerd worden. Maar wanneer er uit het geheugen gelezen moet worden -of er naar toegeschreven moet worden -zal de R800 even moeten 'niksen', het geheugen houdt die snelheid domweg niet bij. In tabel l staat een vergelijking van de executietijden in microseconden.
Het is duidelijk dat de instructies die niets met het geheugen te maken hebben meer versneld worden dan de andere. De 16 bits instructies profiteren het meest: het optellen van twee 16 bits registers doet de R800 bijvoorbeeld 24 keer sneller dan de Z80. Een 8 bits optelling is 10 keer sneller, geheugen-toegang 'slechts' zes keer. Dit levert een gemiddelde versnelling van vier a vijf keer op.

Programmeurskunsten

De turboR is dus ook zonder meer al een stuk rapper dan de 'gewone' MSX, ook de tot zeven Megahertz opgevoerde exemplaren. Maar slimme ML-programmeurs kunnen met de 'paged DRAM' geheugen-truuk nog veel meer tijd winnen, hoewel het programmeren een lastige klus zal zijn.
Die paged DRAM access betekent dat alle toegang tot het geheugen zich in dezelfde 256 bytes moet afspelen. Er is maar één manier om dat te realiseren: een subrouti-ne van minder dan 256 bytes, die begint op een 256-byte grens. De lage byte van het beginadres is dan nul. De stack moet in hetzelfde 256-byte segment liggen, als er tenminste een stack nodig is. Vergeet niet dat er altijd interrupts kunnen optreden die de stack gebruiken! Wie ook daar geen last van wil nebben zal de interrupts uit moeten schakelen.
De gezamenlijke data die door de subroutine gebruikt wordt, moet zich in ook hetzelfde segment bytes bevinden. Code, data, werkruimte en stack mogen samen niet meer dan 256 bytes in beslag nemen. Maar al die beperkingen komen de snelheidswinst ten goede. Reken maar rustig op tussen de tien en twintig keer het tempo van de Z80!

Geheugen opfrissen

Onder andere hierdoor is de timing van de R800 niet constant. Een veelgebruikte manier om korte periodes te meten is, de instructie-timing van de CPU te gebruiken.

De bekende 'wachtlusjes' berusten op dit principe.
Op de R800 gaat dat om verschillende redenen niet goed. Ten eerste kan er net DRAM page access plaatsvinden, waardoor toegang tot het geheugen sneller is. Verder vindt de 'memory refresh' niet meer tussen de processortaken door plaats, maar zo'n 32 keer per seconde. Elke refresh duurt 280 nanoseconden, waardoor de processor gewoon even moet wachten. Toegangstijden tot het geheugen variëren ook van pagina tot pagina: als de S1990-chip - de nieuwe MSX-Engine -pagina's moet omschakelen voor de R800 is geheugentoegang natuurlijk trager.
Om tijden toch nauwkeurig te kunnen meten, is de turboR voorzien van een systeem-timer. Deze loopt onafhankelijk van het systeem met een vaste snelheid.

De instructieset

Maar de R800 draait niet alleen Z80 code: de instructieset is uitgebreid. In het kader bij dit artikel staat een opsomming van alle veranderingen. De meeste verschillen zijn echter cosmetisch en de registerset is dezelfde. Vrijwel alle instructies hebben een andere naam gekregen, maar ze werken precies hetzelfde. Verder zijn de beide index-registers IX en IY nu ook per byte aanspreekbaar, voor sommige instructies althans.

De lage helft van IX heet IXL, de hoge IXH. Op dezelfde manier wordt IY gesplitst in IYL en IYH.

De derde verandering is wel weer een hele belangrijke: de R800 kan vermenigvuldigen. De instructie MULUB vermenigvuldigt het A register met een ander 8 bits register en levert een 16 bits resultaat. MULUW vermenigvuldigt HL met BC of SP en levert een heus 32-bits antwoord op in HL en DE. Daarnaast is er een IN F,(C) instructie bijgekomen, die een byte uit een I/O poort in het vlagregister kan halen. Dat maakt het testen op allerlei bitjes wel heel eenvoudig. Naast het gebruik van de halve indexregisters zijn er dus feitelijk maar drie nieuwe instructies in de R800. Toch zijn er nog wel een aantal instructies te verzinnen waar de bouwers van de R800 de machinetaal programmeurs blij mee zouden hebben kunnen maken. Wat dacht u van dingen als LD (BC),n of LD (DE),HL of PUSH nn? Jammer genoeg zijn dit soort zaken niet toegevoegd.

Tenslotte

Maar het staat als een paal boven water dat de R800 een knap stukje werk is. Niet veel meer mogelijkheden dan de Z80, maar wel een stuk sneller. Dat wil zeggen dat iedereen die nu de Z80 kan programmeren, ook de R800 van programma's kan voorzien. Indien gewenst zelfs met dezelfde assembler. Speciale R800 assem-blers, die de R800 mnemonics begrijpen en de extra instructies ook kunnen vertalen zullen er ook wel komen, maar tot die tijd kan er ook met een Z80 vertaler gewerkt worden. Dat maakt de overstap bijna kinderlijk eenvoudig.

Veranderingen in de instructieset van de R800

Halve index-registers

De LoaD-instructie LD rl ,r2 is nu ook toegestaan met IXL, IXH, IYL en IYH. Het is echter niet mogelijk op deze manier bytes tussen IX en IY uit te wisselen. \
LD A,IXL mag dus, LD EXL,A en LD IXL,IXH mogen ook, maar LD IXLJYL mag niet.
De vier halve index-registers mogen ook gebruikt worden in de volgende instructies:

ADD A,xx:	optellen
ADDC A,xx:	optellen met Carry (was: ADC)
INC xx:	increment
SUB A,xx:	aftrekken
SUBC A,xx:	aftrekken met Carry (was: SBC)
DEC xx:	decrement
AND A,xx:	logische and
OR A,xx:	logische or
XOR A,xx:	logische excusive or
CMP A,xx:	vergelijken (was: CP)

Vermenigvuldiging

De twee nieuwe instructies zijn:
MULUB A,reg, waarbij 'reg' gelijk is aan B, C, D of E. Het 16 bits resultaat van de vermenigvuldiging van het A-register met het genoemde register komt in HL te staan.
MULUW HL,regp, waarbij 'regp' gelijk is aan BC of SP, wat in de praktijk waarschijnlijk altijd BC zal zijn.
Het resultaat komt in HL en DE samen te staan; HL bevat de laagste 16 bits, DE de hoogste.
Deze vermenigvuldiging is 'unsigned', dat wil zeggen: zonder teken. De uitkomst van -l maal -l is dus nonsens, geen +1. Instructies voor 'signed' vermenigvuldiging ontbreken, evenals deel-opdrachten.

INF,(C)
is een nieuwe instructie die een byte van een input-poort leest, waarvan het nummer in het C-register staat. De vlaggen worden gezet naar aanleiding van dit byte, maar het byte zelf wordt niet gebruikt. Handig om een bepaald bit van een poort in de gaten te houden.

Blok verplaatsingen

De 'verplaatst en herhaal'-opdrachten zijn allemaal omgedoopt. Ze zien er 68000-achtig uit, wat de leesbaarheid - voor 68000 kenners wel bevordert.
LDI (LoaD + Increment) is veranderd in MOVE (HL++),(DE++).
Dit betekent zoveel als 'verplaats van (DE) naar (HL), verhoog DE en HL, verlaag BC'. Precies hetzelfde als de oude LDI-instructie, dus.
LDIR (LoaD, Increment + Repeat) heeft een extra M aan het einde van de instructie. Dit staat waarschijnlijk voor 'Multiple':
MOVEM (DE++),(HL++)

Op dezelfde manier zijn de andere twee blokverplaatsingen van naam veranderd:
LDD (LoaD + Decrement):MOVE (DE-).(HL-)

LDDR (LoaD, Decrement + Repeat):MOV EM (DE--),(HL--)

Blok vergelijkingen
Voor deze groep vergelijk-instructies geldt hetzelfde: alleen een naamsverandering.
CPI (ComPare + Increment): CMPA,(HL++)
CPIR (ComPare, Increment + Repeat): CMPM A,(HL++)
CPD (ComPare + Decrement): CMP A,(HL-)
CPDR (ComPare, Decrement + Repeat): CMPM A,(HL~)

Spronginstructies
Ook deze commando's zijn omgedoopt om ze in een moderner jasje te steken. De JP- en JR-instructies hadden een wat vreemde opbouw, waarbij de voorwaarde voor de sprong niet bij de instructie, maar bij de bestemming stond. De voorwaarde komt direct achter de instructie te staan. Daarbij zijn meteen alle 'jump'-opdrachten omgedoopt in 'branch'-opdrachten.

Er zijn twee soorten branch-instructie s: normale en korte. Deze komen direct overeen met respectievelijk JP en JR. BNZ adres komt dus overeen met JP NZ.adres; SHORT BC adres betekent JR C,adres. De DJNZ-opdracht (Decrement + Jump Non Zero) is meteen meegenomen en heet nu DBNZ. Helaas zijn relatieve sprongen - of moeten we nu 'branches' zeggen - nog steeds beperkt tot 129 bytes vooruit en tot 126 bytes achteruit. Een zogenaamde 'long branch' ontbreekt.

IN en OUT
Alweer naamswijzigingen:
OTI (OuT + Increment): OUT (C),(HL++)
OTIR (OuT, Increment + Repeat): OUTM (C),(HL++)
OTD (OuT + Decrement): OUT (C),(HL~)
OTDR (OuT, Decrement + Repeat): OUTM (C),(HL--)
Door OUT in IN te veranderen ontstaan de nieuwe instructies voor het lezen van een I/O-poort.

Accumulator-instructies
DAA: ADJ A
CPL: NOTA
NEG: NEG A
Bij deze instructies moet dus het A-register genoemd worden.

Processor-instructies
CCF: NOTC
SCF: SETC

Verwisselingen
EX: XCH
EXX: XCHX

Schuiven en roteren
RL: ROL
RR: ROR
RLC: ROLC
RRC: RORC
RLA: ROLA
RRA: RORA
RLCA: ROLCA
RRCA: RORCA
RLD: ROLA
RRD: RORA
SLA: SHL of SHLA
SRA: SHR
SRL: SHRA

Bit instructies
RES: CLR

Snelheidsvergelijking R800 en Z80

Instructie	**MSX2**	**R800**	**Versnelling**
LD r1,r2	1.40	0. 14	10.0
LD r,(HL)	2.23	0.4Z	5.3
LD r,(IX+d)	5.87	0.70	8.4
ADD A,r1	1.40	0.14	10. 0
INC r	1.40	0.14	10.0
PUSH rr	3.35	8.56	6. 0
ADD HL, rr	3.35	0. 14	24.0
INC rr	1.96	0. 14	14.0
JP	3.07	0.42	7.3
JR	3.63	0.42	8.7
CALL	5.03	0.84	6.0
RET	3.07	0.56	5.5
LDIR	6.43	0.98	6.6
DJNZ	3.91	B.4Z	9.3
MULUB		1.96	
MULUW		5.03	