

IRQ, I/O, Minneadresser og DMA

En innføring beregnet på elever ved Oslo By Steinerskole.

© Hans Poppe 2001

IRQ

Tidligere har vi snakket om CPU (The Central Processing Unit) og at den består av 2 hoveddeler; ALU (Arithmetic Logical Unit) og kontrollenheten. Vi så på hvordan kontrollenheten ”spilte flasketuten peker på” og gav ordre til ALU og inn/ut enhetene om hvilken enhet som skal ha ALUs oppmerksomhet akkurat i øyeblikket. Nå er det ikke slik at denne ”flasketuten peker på” leken virkelig er som en flasketut som spinner rundt. Det ligner litt, men med en radikal forskjell. Ikke alle som sitter i sirkelen rundt flasken er med å spille hele tiden. Det er som om de av og til rekker en hånd i været, og da er de med, og andre ganger legger de hånden ned i fanget, og da er de ikke med i spillet. Det finnes fire forskjellige måter å rekke hånden i været på. Den ene heter Interrupt ReQuest (IRQ). En IRQ er en ledning som går fra ekspansjonsenheten (for eksempel et lydkort) til kontrollenheten i CPU. Når ekspansjonskortet forstår at det trenger hjelp av ALU til å beregne et-eller-annet, vil det sende en IRQ til kontrollenheten og be om å få prosessortid. (Akkurat som når elevene i klassen rekker hånden i været). En gang i blant vil du få oppmerksomhet med en gang, andre ganger må du vente litt. (Akkurat som når elevene i klassen rekker hånden i været, noen ganger får du ordet med en gang, noen ganger må du vente litt). Denne ledningen går i selve hovedkortet, mellom bussen (for eksempel PCI) og kontrollenheten og ser ut som en tynn strek på den grønne platen. Det er 16 slike ledninger, eller IRQ-er. Disse har nummer fra 0 til 15. Det er en standard for hvilke IRQ som skal betjene hvilke ekspansjonsenheter.

Under vanlig bruk av maskinen vil de forskjellige I/O kortene (inn / ut enhetene) av og til ha behov for oppmerksomhet fra prosessoren. De har et eller annet som må regnes ut. (Mange I/= kort har mulighet til å henvende seg på en annen måte, nemlig ved å bli sett på som minneadresser; dette kommer litt senere i kompendiet). De har muligheten til å avbryte (eng: to interrupt) prosessoren mens den holder på med et program eller en annen oppgave. Dette gjør ekspansjonskortet ved å sende et systemkall av typen interrupt (= INT) eller Interrupt request (= INTR) som avbryter prosessoren og får den til å begynne å gjøre noe annet.

Det vil si, det er ikke fullt så enkelt...

Det som skjer er at de I/O enhetene som trenger oppmerksomhet fra CPU vil sende et IRQ og håpe på å få prosessortid. I midlertid kan det hende at prosessoren holder på med noe som er ENDA viktigere. Da vil den ikke avbrytes.... Derfor vil alltid prosessoren, når den mottar en IRQ (en INT request), sende en INTA (Interrupt request Acknowledge) og derved si fra at den har forstått ønsket, men du er plassert i en kø. Du vil bli betjent når det er din tur.... Så dukker det opp en annen situasjon, hva hvis to I/O enheter sender en INT samtidig? Hvem blir da betjent først? Her kommer IRQ numrene inn. Som nevnt ovenfor er de nummerert fra 0 til 15 (altså 16 stk). Jo lavere tallet er, jo høyere prioritet har IRQ-en. Hvis et lydkort (med for eksempel IRQ 5) og et nettverkskort (med for eksempel IRQ 9) sender en INT helt nøyaktig samtidig vil den med høyest prioritet (lavest numerisk verdi) bli betjent først. Denne køordningen er viktig, ellers ville prosessoren få flere oppgaver samtidig og det greier den

ikke. Etter at prosessoren er ferdig med å behandle denne INT-en vil den gå tilbake til det programmet den holdt på med og fortsette arbeidet med det.

Bare for å gjøre dette litt vanskeligere; det er to forskjellige typer interrupts som brukes i microcomputere. Jada, det er en microcomputer du bruker, selv om du sikkert synes den er ganske så kraftig ☺. La oss se på dem:

- Maskable interrupts (MI), som computeren, i visse tilfelle, kan velge å ignorere.
- Non-maskable interrupts (NMI), som prosessoren ALLTID MÅ svare på.

Dette foregår slik at microprosessoren har en ”IRQ ut-linje” (kalles INTE som står for INTerruptEnable) i tillegg til de 16 IRQ inn-linjene den skal betjene. Denne linjen brukes til å fortelle de forskjellige ekspansjonseenhetene at den tillater avbrudd, og signalet i INTE (det er et logisk signalnivå, altså den sender ut et tall med en verdi og denne verdien avgjør om INT blir akseptert eller ikke). Denne INTE verdien kan vanligvis bestemmes av software, noe som selvfølgelig betyr at det er programmet som allerede kjører på prosessoren som avgjør om det vil la seg avbryte eller ikke. Dette høres jo kult ut, PROGRAMS RULES!! Neida, bare delvis. Det er noen av IRQ-linjene som ikke vil la seg forstyrre av slikt ”INTE” tull og tøys... Disse vil alltid avbryte prosessoren.

For å forklare dette en gang til. litt kortere:

Prøv å tenke på en programmerbar interrupt controller (en integrert krets eller IC fra det engelske ordet integrated circuit) og hvilket forhold den egentlig har til CPU. Interrupt controller chipen aksepterer avbruddsønsker (IRQ) etter et prioriteringssystem fra inntil 8 IRQ-kanaler, nummerert fra 0 til 7. Når en periferienhet ønsker oppmerksomhet fra kontrolleren vil det sende et signal på en IRQ linje til kontrolleren. Kontrolleren svarer ved å sende et INT til prosessoren. Hvis det kommer to interruptrequests samtidig vil den med høyest prioritet bli behandlet (betjent) først. Jo lavere nummer IRQ-en har, jo høyere prioritet har den. Altså, et signal fra IRQ 0 vil bli behandlet før et IRQ 7 signal.

Her er det, for de skarpeste, noe veldig rart.... Vi snakker om 0 til 7, men vi sa innledningsvis at det er 0 til 15 IRQ-er.

Dette er egentlig noe som er ”jukset til”. Den gangen da det satt en gjeng med gutter på et gutterom et sted i USA og ”fant opp” PC-en. De mente at 8 IRQ var nok og gjorde dette til et 8 bits system. Senere kom det for en dag at dette var litt lite og man laget et system hvor IRQ nr. 2 forfalskes og får nye IRQ-undernumre. Man sier at IRQ2 går i kaskade til IRQ 9. Dette betyr at alle IRQ numre som er høyere enn 7, egentlig blir sendt til IRQ2 som sender videre til 9, 10, 11, 12, 13, 14 og 15.

IRQ	Device
0	System timer
1	Keyboard controller
2	Video graphics adapter (EGA/VGA)
3	Second serial port (COM2, COM4) or bus mouse
4	First serial port (COM1, COM3)
5	Second parallel port (LPT2, often sound card)
6	Floppy disk controller
7	First parallel port (LPT1)
8	Real time clock
9	Redirect to IRQ 2 (often network card)
10	Available (often soundcard)
11	Available
12	PS/2 Mouse (serial mouse uses comport)
13	Math coprocessor
14	Hard disk controller
15	Available (often network card or SCSI controller or secondary harddisk controller)

I/O – adresser

I/O adresser, eller I/O porter brukes for at programmer skal kunne kommunisere med hardware som for eksempel et nettverkskort. I/O porter kan lettest sammenlignes med toveis radiosignaler. Disse går i samme medium (luften) uten at de forstyrrer hverandre. Det gjør de fordi de bruker forskjellige frekvenser. Man kan godt trekke en parallell mellom disse frekvensene og I/O-porter. Det er vanlig i en PC at det er tusenvis av I/O-porter (adresser) og det er normalt ikke et problemskapende felt. Det vil oppstå konflikter hvis to ekspansjonsenheter forsøker å snakke til et program via samme I/O-port. Hvis dette skjer må vi manuelt endre I/O-adressen, men det er ikke slik som med IRQ, at det er vanskelig å finne en ledig adresse. Det finnes, som sagt, vanligvis tusenvis av slike adresser. Bare bytt den! Dette gjøres i kontrollpanel => systemegenskaper. Hvis du skulle komme ut for gamle kort, vil det ofte være en jumper som må stilles om for å endre både IRQ og I/= -adresser. På nyere kort vil dette vanligvis kunne gjøres med software. (OBS! hvis du ser på dette i kontrollpanelet vil en del kort bare oppgi hvilken I/O-adresse de starter med, ikke nødvendigvis alle det bruker.

I/O er den klassiske måten å sende data til og fra CPU. Det involverer både ALU og kontrollenheten.

DMA – Direct Memory Access.

DMA tillater høyhastighetsoverføring mellom ekspansjonsenheten og minne, UTEN at prosessoren er involvert. Dette forutsetter at ekspansjonskortet har en egen form for prosessering. Dette gjør at kortet kan operere raskere.

Det er bare nyere kort som har DMA egenskaper. I likhet med I/O har DMA en slags adresser, men disse kalles kanaler (DMA kanal 0 til 7, altså 8 stk). Det er en tilsvarende kaskadefunksjon mellom DMA 1 og DMA 4.

Det er viktig å huske på at DMA er kult fordi det går raskt, men det ukult fordi det går direkte. Årsaken til at det er ukult er rett og slett fordi prosessoren og kontrollenheten ikke vil få anledning til å kontrollere data for feil (paritetskontroll). Derfor er det lettere for dårlig software å få maskinen til å krasje.

DMA kanalene:

DMA kanal #	Funksjon (standard)	Bus slot?	Korttype	Overføring	Anbefalt bruk
0	Ledig	Ja	16 bit	8 bit	integrert lyd
1	Ledig	Ja	8/16 bit	8 bit	8 bits lyd
2	Floppydisk kontroller	Ja	8/16 bit	8 bit	Floppy kontroller
3	Ledig	Ja	8/16 bit	8 bit	LPT1 (i ECP modus)
4	Kaskade til DMA 1	Nei	---	16 bit	---
5	Ledig	Ja	16 bit	16 bit	16 bit lyd
6	Ledig	Ja	16 bit	16 bit	ISA SCSI adapter
7	Ledig	Ja	16 bit	16 bit	Ledig

Helt til slutt skal vi se raskt på hvordan dette egentlig ser ut på vår egen maskin.
Minimer alle vinduer => høyreklikk på "Min datamaskin" => velg "behandle" => ekspander "systemegenskaper" => ekspander "maskinvareressurser" og sjekk alle 6 undermapper. Hvordan ser det ut?

Minneadresser:

Det er ikke så mange ekspansjonskort som bruker minneadresser, det vil si et reservert område i RAM som bare det får lov til å bruke.

Det er vanligst at det er skjermkortet og såkalt "firmware" det vil si BIOS som får bruke reservert minne. Listen over kort som får bruke minneadresser er ikke veldig lang, men den er ikke uinteressant. Ingen får lov til å bruke samme minneadresse som noen andre!!

Her følger en tabell over de reserverte adressene i området 640K til 1MB av RAM:

Device	Adresserange i Hexadesimal	Anmerkning
Graphics Mode Video RAM	0A0000-0AFFFF	
monokrom text modus video RAM	0B0000-0B7FFF	
Farve textmodus video RAM	0B8000-0BFFFF	
Video RAM for VGA / super VGA	0C0000-0C7FFF	
Ledig	0C8000-0DFFFF	Tilgjengelig for BIOS utvidelseskort eller minnebehandlere slik som QEMM eller EMM386
Hovedkortets ROM	0E0000-0EFFFF	Regnes som ledig dersom BIOS ikke bruker den (ikke alle BIOS trenger dette)
Hovedkortets ROM, alle systemer:	0F0000-0FFFF	