1.4 Interrupts

Friday, March 20, 2020 6:02 PM

Praktisk talt alle computere har en avbruddsmekanisme hvor andre moduler (e.g. I/O, minne) kan avbryte en normal sekvensering av prosessoren. Det finnes ulike typer avbrudd:

1. **Program:** Generert av en tilstand som oppstår som et resultat av en instruksjonsutførelse (execution), som for eksempel aritmetisk overflow, deling på null, forsøk på å utføre en ulovlig maskin instruksjon, eller en referanse utenfor brukerens tillatte minneplass.
2. **Timer:** Generert av en timer inni prosessoren. Dette tillater operativsystemet å utføre noen funksjoner på generelt basis.
3. **I/O:** Generert av en I/O-kontroller til å signalisere normal gjennomføring av en operasjon eller til å signalisere en variasjon av error-tilstander.
4. **Hardware-feiling:** Generert av en feil, som for eksempel strømtap eller minne parity error.

Avbruddsmekanismer har som hovedfunksjonalitet at de kan forbedre prosessorutnyttelsen. For eksempel er I/O-enheter mye tregere enn prosessorer. Uten avbrudd kan dette sinke utnyttelsen av prosessoren, ettersom den må vente og forbli inaktiv inntil I/O-enheten blir ferdig.

Med avbrudd kan dermed prosessorer brukes til å utføre andre instruksjoner mens de venter på eksempelvis I/O-enheten til å bli ferdig. Dette innebærer en implementasjon av en avbrudds-håndterer (interrupt-handler) - dette er generelt en del av operativsystemet. Den eksterne enheten sender et signal (interrupt request) til prosessoren, hvorpå prosessoren responderer med å utsette operasjonen i det kjørende programmet. Dette håndteres så av avbrudds-håndtereren. Det legges til et avbrudd-stadiet i instruksjonssykelen for å si om det oppstod noen avbrudd underveis.

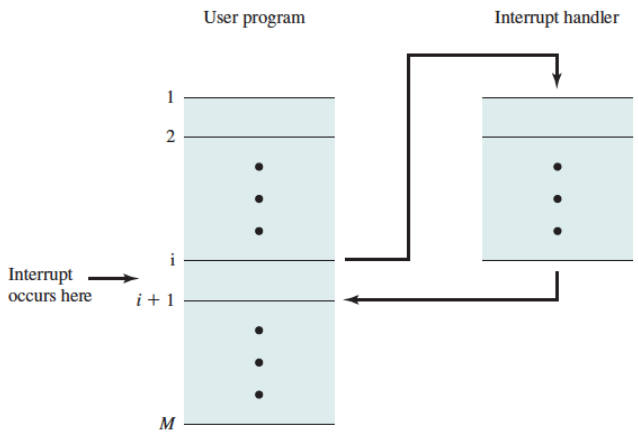

Figure 1.6 Transfer of Control via Interrupts

Avbruddsprosessen (Interrupt Processing)

Et avbrudd trigger flere hendelser, både i prosessor hardware og i software. Når en I/O-enhet gjennomfører en I/O-operasjon skjer følgende hardware hendelser:

1. Enheten sender et interrupt/avbrudds-signal til prosessoren.
2. Prosessoren fullfører execution/handlingen til den nåværende instruksjonen før den svarer på avbruddet.
3. Prosessoren tester for en ventende avbruddsforespørsel, avgjør om det finnes en, og sender så et acknowledgement-signal (ACK, anerkjenner) til enheten som sendte avbruddet. Dette ACK-signalet tillater enheten å fjerne sitt avbruddssignal.
4. Deretter må prosessoren forberede seg for overføringskontroll til avbruddsrutinen. I starten lagrer den informasjonen den trenger for å kunne fortsette det kjørende programmet ved tidspunktet til avbruddet. Den minimale informasjonen påkrevd er programmets status ord (PSW, inneholder statusinformasjon om den daværende kjørende prosessen, inkludert minnebruksinformasjon, tilstandskode og annet statusinformasjon som en avbrudds-av-og-på bit, samt en kernel/bruker-modus-bit.) og lokasjonen til den neste instruksjonen som skal utføres, som befinner seg i PC. Disse kan pushes til en kontroll-stakk (control stack).
5. Prosessoren laster så PC-en med inngangslokasjonen til avbruddshåndteringsrutinen som skal svare på dette avbruddet. Avhengig av computerarkitekturen og OS designet, kan det være et enkelt program, en for hver ulik type avbrudd, eller en for hver enhet og hver type avbrudd.

   Når PC-en har blitt lastet inn, fortsetter prosessoren til den neste intruksjonssykelen, som da starter med en instruksjons-fetch. På grunn av at instruksjons-fetchen avhenger av innholdet til PC, blir kontroll (control) sendt til interrupt-handler-programmet. Execution av dette programmet resulterer i de følgende operasjonene:

6. På dette tidspunktet er PC-en og PSW-en tilhørende det avbrutte programmet blitt lagret i control stack. I tillegg til dette må mer informasjon lagres, blant annet innholdet til prosessorregistrene,

ettersom disse kanskje blir brukt av interrupt-handleren. Etter all den nødvendige informasjonen er lagt til i control stack oppdateres stack-pekeren til det øverste elementet, og PC-en oppdateres til å peke på starten av interrupt-service-rutinen.
7. Etter hvert må interrupt-handleren prosessere avbruddet. Dette inkluderer en undersøkelse av statusinformasjonen relatert til I/O-operasjonen eller andre hendelser som forårsaket avbruddet. Det kan også innebære å sende en ekstra kommando eller ACK-signal til I/O-enheten.
8. Når avbruddsprosessering er fullført, hentes de lagrede registerverdiene fra control stack og blir gjenopprettet til registrene.
9. Det siste steget innebærer å gjenopprette PSW-en og PC-ens verdier fra control stack. Dette muliggjør at den neste instruksjonen som skal utføres er fra det tidligere avbrutte programmet.

Flere avbrudd (Multiple interrupts)

Det er mulig at det oppstår flere avbrudd i et kjørende program; for eksempel hvis programmet får tilsendt data fra en kommunikasjonslinje og skal så printe resultatene samtidig.

Det er to tilnærminger for å håndtere en slik situasjon:

1. **"Førstemann-til-mølla" (*disabled interrupt)***
   I dette tilfellet blir avbrudd avslått dersom det allerede er et avbrudd som blir prosessert. Det vil si at prosessoren sjekker for om det har oppstått noen avbrudd etter den har fullført prosesseringen av det første avbruddet, altså når interrupt-handler-rutinen er fullført. Dette er en enkel måte å håndtere flere avbrudd på, hvor alle interrupts håndteres i en strent sekvensiell rekkefølge.

   - Ulemper:
     Denne håndteringen tar ikke hensyn til relativ prioritet og tidskritiske behov. Eksempelvis kan det hendte at input som kommer fra en kommunikasjonslinje må blir absorbert raskt for å gi plass til mer input. Dersom den første delen av input ikke har blitt prosessert før den andre delen kommer kan det hende data blir borte som en konsekvens av at bufferet til I/O-enheten blir overfylt og overflow.

2. **"Prioritetsrekkefølge"**
   Denne tilnærmingen innfører en idé om en prioritetsrekkefølge der noen enheter har høyere prioritet enn andre, og dermed kommer først i køen dersom de sender et interrupt-signal. I praksis vil dette si at dersom en enhet med lavere prioritet er midt i interrupt-handling-rutinen sin, så kan en enhet med høyere prioritet "trumfe" den andre enheten dersom den sender ut et interrupt-signal. I så fall lagres statusinformasjonen og tilstandsinformasjonen til den første enheten i en control stack, og fortsetter først når den andre enheten er ferdig med sin interrupt-handling-rutine.