# Hoofdstuk 2

## Instructies

Programma voert instructies uit;

Gewone instructies: mov, add, imul, idiv, jmp, cmp

Gepriviligeerde instructies: hlr, in, out

## Processortoestand:

Veiligheid!!!

### User mode:

Standaard modus, heeft toegang tot de gewone instructies.

### Kernel mode:

Heeft toegang tot alle instructies, ook de I/O instructies enkel het besturingsysteem kan in kernelmode gaan.

## Interrupts

Processor voert één programma tegelijk uit, het bestuuringssysteem is een programma.

Besturingsysteem moet in actie komen bij een I/O input of vraag van hogergelegen software

OS is event driven

* Altijd in het geheugen
* Enkel actief wanneer nodig

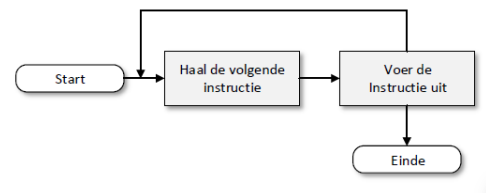
### Interrupt requests:

Stel: netwerkpakket komt binnen

1. Netwerkkaart stuur interrupt request.
2. Processor onderbreekt het actieve programma

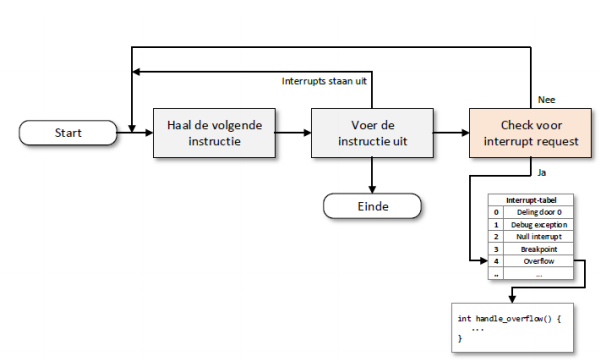
Geen interrupt requests = geen onderbreking van het actieve programma.

### Haal- en uitvoeringscyclus:



1. Haal de volgende instructive
2. Voer de instructive uit
3. Als er nog een instructive is, begin opnieuw
4. Al ser geen meer is einde.

### Met interrupt ondersteuning:



1. Haal de volgende instructive
2. Voer instructive uit
3. Check voor interrupt request
4. Er is geen interrupt: begin terug opnieuw
5. Er is wel een interrupt, kijk welke in de interrupt table.
6. Handel interrupt

Afhandeling van interrupts kan worden uitgeschakeld (tijdens het afhandelen van een interrupt)

Interrupt request = elektrisch signal

Veel mogelijke redenen:

* Delen door nul
* Leesopdracht voltooit
* …

Elke interrupt moet anders afgehandeld worden

### Interrupttabel

Voor elke interrupt een geheugenadres (interrupt vector)

#### Voorbeeld: leesopdracht harde schijf

1. Schijfbesturingseenheid ontvangt een leesopdracht
2. Leesopdracht is voltooid => interrupt request naar interrupt controller.
3. Interrupt controller geeft request door aan processor
4. Interrupt controller sturrt volgnummer van de oorsprong van de interrupt request
5. Processor zoekt adres van de juiste interrupt handler in de interrupttabel
6. Inhoud van de bevelenteller, registers en de vlaggen worden opgeslagen.
7. Adres van interrupt handler wordt in bevelenteller geplaatst en de processor wisselt naar Kerneltoestand
8. Sprong naar handler, en de request wordt afgehandeld
9. Interrupt handler herstelt bevelenteller, registers en processorstatus en de processor wisselt aar gebruikerstoestand.

stap 6 en 7 zijn de context switch toestand wijzigen van de processor

* Toestand van huidige programma opslaan
* Toestand van nieuwe programma laden
* Volgende instructive uitvoeren

## Soorten interrupts

### Externe interrupts (= van buiten de processor)

* Klok inerrupts
* I/O interrupts
* Keyboard interrupts
* Hardwarefouten

Zijn asynchroon, hebben niets te maken met huidige programma

### Interne interrupts (= interrupts van binnen in de processor)

* Uitzonderingen (exceptions)
* Traps (software interrupts)

Zijn synchroon, toepassing kan niet verder tot dat de interrupt afgehandeld is

## Interrupt handler:

Ineterne interrupts:

Processor gaat conditiecode in toestandsbeschrijving na

Programma dat trapt naar BS geeft informatie mee om juiste handler te kiezen

Externe interrupts:

1. Langs de in interruptlijn, afhankelijk van de interruptlijn waar het interrupt signal binnenkomt (niet meer controllers mogelijk dan interruptlijnen)
2. Gemeenschappelijke interruptlijn, Software pollen welke controller een interrupt heft gegeven (traag)
3. Gemeenschappelijke interruptlijn en acknowledge lijn, gaat langs deze lijn na welke controller een interrupt heft gestuurd, Daisy Chaining (snel langs hardware)

### Meerder interrupts tegelijk?

Welke eerst?

Prioriteit, afhankelijk van de prioriteit bepaalt door de interrupt handler.

Asynchronen interrupts tijdens interrupt afhandeling kunnen tijdelijk worden geblokeerd of als ze een hogere prioriteit hebben dan de huidige uitvoeren.

## System calls

Interrupts, Hardware <> besturingssysteem

System calls, Software <> besturingssysteem

* Opstarten of beeindigen van een programma
* Programma laten wachten op een gebeurtenis
* Toekennne en vrijgeven van geheugen
* Bestandsbeheer
* Besturing en beheer randapparaen
* Uitisseln van informatie

### Read-system call in Linux

Read (file\_desc, buffer, nbytes) = C code, wanneer een programma deze functie aanroept zal hetvolgende gebeuren:

1. Oproepend programma zet warden van de drie parameters op de staple
2. Read functie wordt aangeroepen
3. Functie zet system call nummer in het juiste register en voert TRAP functie uit
4. Interrupt service routinge van het OS zoekt juiste routine op voor de system call
5. Processor start systeemroutine
6. Systeemroutine zet resultaat op voorziene plaats, verlaat kernel mode en geeft controle terug aan de read functie
7. Read functie ontvangt gegens
8. Uitvoer oproepend programma gaat verder

# Hoofdstuk 3: boot process

## Opstarten:

1. Voiding inschakelen

* Eigen spanningsniveaus controleren
* Stabiele spanning? (signal naar moederbord)

1. Moederbord

* Continu rest-signaal naar processor
* Power good? Reset signal valt weg

1. Processor start

* Kernel mode
* Adressering in Real mode
* Op adres 0xFFFFFF0

1. Memory-mapped I/O

* Inhoud van de ROM-chip wordt gemapped op de RAM

1. Opstarcode UEFI (Unifed Extensivle Firmware Interface) wordt geladen

* Mini-besturingssysteem dat de hardware initialiseert en het besturingsysteem inlaadt
* Opvolger van BIOS
* Modulair ontwerp
* Boot service: enkel voor BS is opgestart (UEFI interface)
* Runtime service: altijd besikbaar (opvragen datum/tijd)

1. Besturingsysteem inladen (Partition bootloader)

* Besturingsysteem wordt van de harde schijf gehaald (primary partition)

1. Boot loader
2. BOOTMGR

## Real vs. protected mode

### Real mode:

* 20-bit adressen
* In segmentregister

### Protected Mode:

* 32-bit adressen
* Basis in segmentdescriptor