MASKINVARE

BEGREPER/FORKLARINGER

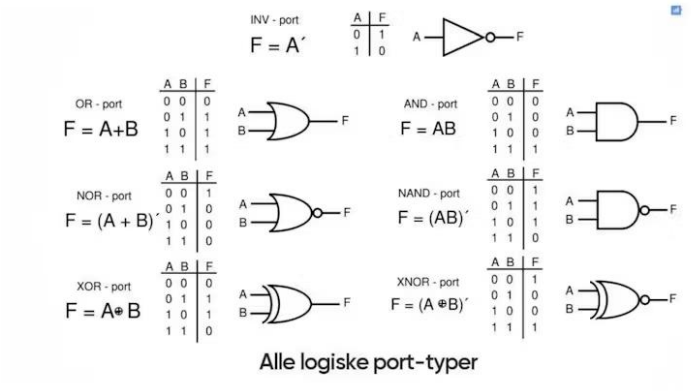
IBM 650: Maskin laget av IBM på 50-tallet for å løse vitenskapelige beregninger og bistå i forretning

Registre: lagerlokasjoner helt tett på prosessoren (de er på 64 bit, altså 8 byte)

Flagg: 1-bits lagre som settes automatisk av prosessoren. Flagg er 0 eller 1 avhengig av hva som skjer i prosessoren

Porter

- Har alltid en logisk funksjon
- Kan ha matematisk funksjon
- Et hvilket som helst boolsk funksjonsuttrykk kan implementeres med bare NAND eller bare NOR
    - Vi foretrekker NAND
- Logiske porter: er elektroniske kretser som utfører grunnleggende logiske funksjoner ved å ta inn to binære innganger og gi en binær utgang basert på en fastsatt logisk regel (som AND, OR, NOT, NAND, NOR, XOR, XNOR)
    - En logisk funksjon er en operasjon som tar inn binære innganger og produserer en binær utgang basert på en bestemt sannhetstabell
    - En sannhetstabell lister opp utgangene av en logisk operasjon for alle mulige innganger



Funksjon: kan være en logisk eller matematisk (XOR, XNOR, AND) funksjon. Én funksjon inneholder en eller flere porter

Minterm

- En funksjon kan være gitt på «sum av produkt» form
    - Eks. F = xy + xy' + x
- Hvert «produktledd» som inneholder alle variablene kalles **minterm**
- For to variabler finnes det 2^2 forskjellige mintermer, for tre variabler finnes det 2^3 forskjellige mintermer

- xy + x'y + xy' + x'y'
- Notasjon: m_x

Maksterm

- En funksjon kan være gitt på «produkt av sum» form
  - Eks. F = (x + y) (x + y') y
- Hvert «summenledd» som inneholder alle variablene kalles **maksterm**
- For to variabler finnes det 2^2 forskjellige makstermer
  - (x + y) (x' + y) (x + y') (x' + y')
- Notasjon: M_x

Binær adder

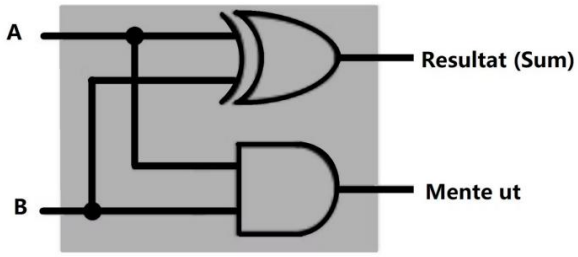
- Vanlige anvendelser: mikroprosessor ALU, Xbox, mikserbord, digitalt kommunikasjonsutstyr etc.
- Basis for addisjon, subtraksjon, multiplikasjon, divisjon og mange andre matematiske operasjoner
- All form for filtrering/signalbehandling

Binær addisjon

0 + 0 = 0

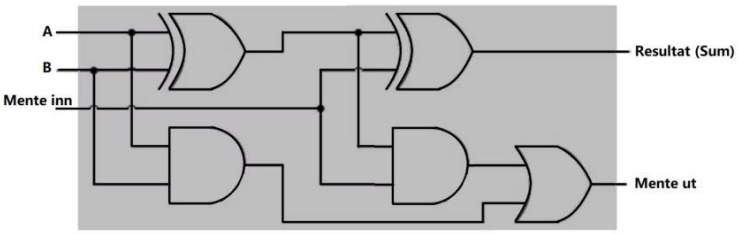
1 + 0 = 1

1 + 1 = 10 (svaret blir 0, med 1 i mente)



HALV-ADDER (HA)

Halvadder: tar ingen mente inn. Den har to innganger og to utganger (halvadder har to inputs og returnerer en sum og mente). Den adderer de to minste signifikante bittene.



1-bits fulladder

Fulladder: adderer sammen bit A_n og B_n med eventuelt mente inn. Den har tre innganger og to utganger (fulladder har tre inputs, hvorav det ene inputet er mente, og den returnerer sum og mente). En fulladder består av to halv-addere og flere porter.

Sannhetsverditabell

- Funksjonsbeskrivelse
- Inneholder alle mulige kombinasjoner av innganger
- Én sannhetsverditabell kan ha mange funksjonsbeskrivelser
- Én funksjon har bare én sannhetsverditabell

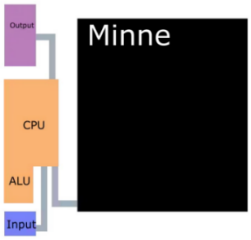
A	B	F
0	0	
0	1	
1	0	
1	1	

Komparatorer: brukes for å bestemme ting, skal vi gå den veien eller den. Nyttig krets for å sammenligne to bits, er den større eller mindre.

Multiplekser

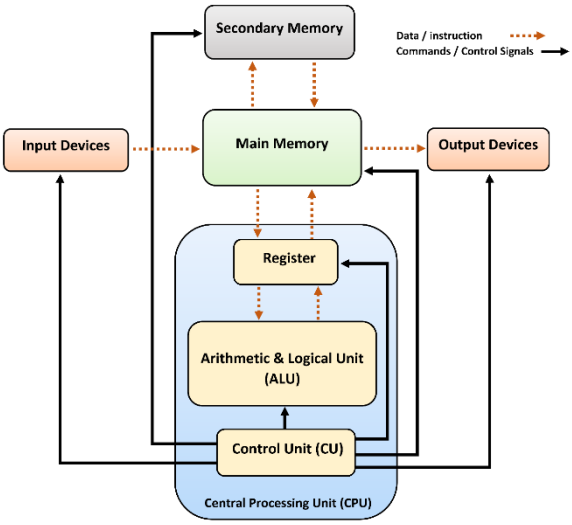
- Nyttig krets i design, gjør at du er i stand til å velge hvilket inngangssignal du vil se på
- En inngang som velger hvilken av de andre inngangene den vil slippe gjennom til utgangen
- Demultiplekser: velger hvilken av utgangene den skal sende signalet til

CPU: «hjernen» i en datamaskin. CPU'en utfører de fleste av beregningene som er nødvendige for å kjøre programmer, og håndterer dataene som strømmer gjennom systemet. Hovedkomponentene i en CPU er CU (kontrollenhet), ALU og registre. CPU-en følger en syklus kjent som instruksjonssyklusen eller fetch-decode-execute-syklusen:



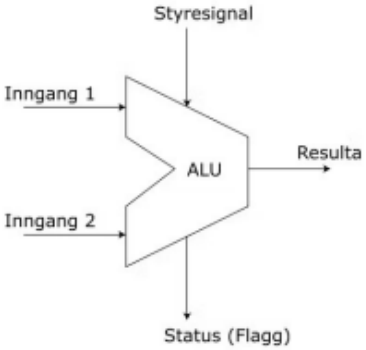
LMC - forenklet

- Fetch: Henter instruksjonen som skal utføres fra hovedminnet.
- Decode: Dekoder instruksjonen for å bestemme hvilke handlinger som kreves.
- Execute: Utfører den nødvendige operasjonen ved å bruke ALU, registre og andre komponenter.
- Store: Lagrer resultatet av operasjonen tilbake i minnet eller registrene hvis nødvendig.



Aritmetisk logisk enhet (ALU): en grunnleggende komponent i en datamaskins sentralenhet (CPU). Den utfører aritmetiske og logiske operasjoner på binære tall, som er essensielle for databehandlingen. CU gir instruksjoner til ALU om hvilken operasjon som skal utføres. ALU gjør alle de grunnleggende operasjonene som datamaskinen trenger for å kjøre programmer.

Styresignal: Elektrisk signal som brukes til å kontrollere oppførselen eller operasjonen til en krets, komponent eller system i en datamaskin. For å avgjøre hvor mange biter styresignalet må være på må vi finne antall unike kombinasjoner som er nødvendig for å velge mellom operasjonene i ALU-en. Hver operasjon må ha en unik kode, og antall biter i styresignalet bestemmes av antall operasjoner som ALU-en skal utføre. Hvis vi bruker eksempelet:



- En ALU som skal gjøre følgende operasjoner: addere, NAND, XOR, NOR, XNOR
- Dette gir totalt 5 operasjoner.
- Antall biter (n) i styresignalet bestemmes av antall kombinasjoner som kan representeres med n biter. Hver bit kan være enten 0 eller 1, og totalt antall kombinasjoner med n biter er derfor: 2^n

- Med n = 2 kan du representere 4 kombinasjoner, altså ikke nok til å dekke antall operasjoner i dette eksempelet.
- Med n = 3 kan du representere 8 kombinasjoner, og det er det nærmeste tallet vi kommer til 5, så det betyr at 3 biter i styresignalet er nok til å utføre de fem operasjonene som skal utføres.

BUS: kommunikasjonskanalene mellom de ulike elementene, mellom registre (ALU, minne og input/outputenheter). Kan deles mellom flere enheter, men kun én som kan sende av gangen. Internt i en CPU er det én eller flere bus(er).

Flipflop: fliflop / vibbe er raskest minne element. Vi bruker flipflops fordi vi vil ha et stabilt signal gjennom en klokkesykel. Det er for at komponentene i kretsen skal kjøre synkronisert. Det finnes ulike typer flipfloper:

- D-flipflop låser Q (utgangsverdien) til D (inngangsverdien) på hver positive klokkeflanke
- JK-flipflop har to inngangsverdier J og K, der J setter Q til 1 og K setter Q til 0. Når både J og K er 0 er Q låst, og når begge er 1 inverteres Q
- T-flipflop er det samme som JK-flipflop, bare at det er inngang koblet til både J og K. Når T er 1 inverteres Q

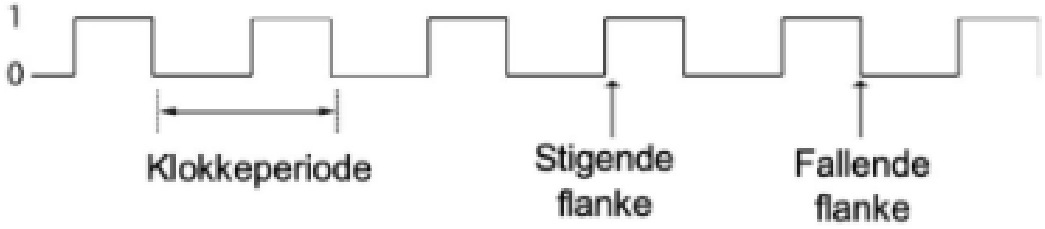
Moores lov: dobling av antall transistorer hvert 1,5. år / 2. år

Transistorer

- Transistorer er laget av silisium og oksygen -> silisiumoksid (sand/glass)
- Transistorer kan kun representere to tilstander: 0 og 1
- En inverter inneholder to transistorer
- Transistorene er komplementære
- Sammenhengen mellom logiske porter og transistorere er at logiske porter er implementert ved hjelp av transistorer
    - For eksempel, en AND-port i en digital krets kan lages ved å koble transistorer på en slik måte at strømmen kun kan flyte gjennom kretsen
- De ulike transistorene kan kobles sammen

Klokkesykel: grunnleggende tidssyklus i en datamaskins prosessor som synkroniserer operasjonene av de interne komponentene. Representerer én full svingning av klokkesignalet, som går fra lav (0) til høy (1) og tilbake til lav (0) igjen

Klokkeperiode: den faktiske varigheten av én fullstendig klokkesykel



Én klokkesykels CPU

- Klokkeperioden må være like lang som den lengdte forsinkelsen for enhver instruksjon gjennom data-path'en
- Resultat: Raskere instruksjoner tvinges til å bruke mer tid enn nødvendig
- Mulig løsning: Klokke med variabel periode
- Vurdering: Vanskelig å implementere

Klokkesignal: et klokkesignal er et digitalt signal som veksler mellom 0 og 1 med fast takt. Klokkesignalet synkroniserer operasjonene i en CPU

Klokkefrekvens: måles i hertz (Hz), og representerer antall klokkesykluser per sekund. Jo høyere klokkefrekvens, desto flere operasjoner kan prosessoren potensielt utføre i løpet av en gitt tidsperiode (kan bety raskere ytelse). Frekvens = 1 / klokkeperioden

Synkronisering: global klokkesignal – gjelder for hele CPU-en, lokale klokkesignaler – kan ha ulike hastigheter

ISA (kode) er hvordan instruksjoner sender/distribueres i en datamaskin

- Sett med regler på hvordan hardware skal løse et problem og oppføre seg
- Våre instruksjoner blir oversatt til noe maskinen kan se og gjøre
- Mikroarkitekturen er måten datamaskinen er bygget opp på inni for å gjøre jobben sin

Utførelse av addisjon (instruksjon 199)

- Hent instruksjonen og øk programtelleren (FETCH)
- Splitt instruksjonen og hent fra minne (DECODE)
- Utføre instruksjonen (EXECUTE)
- Lagre data (WRITE-BACK)

Micro Architecture (hardware, datamaskinarkitektur)

Pipeline

- Metode/teknikk for å effektivisere ting
- Optimaliseringsmetode
- Samlebåndsprinsipp
- Dele opp instruksjonen i uavhengige deler
- Neste instruksjon kan settes i gang før forrige er ferdig
    - Men hvert ledd er uavhengig
    - Ideelt sett og aller helst skal leddene ta like lang tid
    - Den tregeste delen av prosessen setter hastigheten
- Pipeline med subinstruksjoner
    - IF – Instruction Fetch
    - DE – Decode Instruction
    - EX – Execute
    - WB – Write Back
- Det å ha en 4 trinns pipeline betyr ikke at man får 4 ganger raskere prosessering. Det går alltid noe tid bort til administrering av instruksjoner og komplikasjoner.

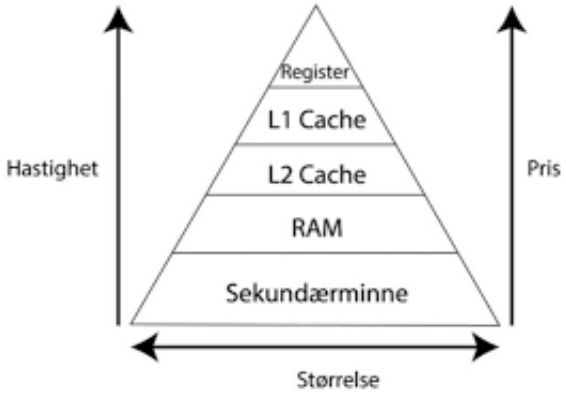
Minnehierarki

Register, cache og RAM har flyktig minne (minnet slettes når strømmen skrus av), i motsetning til harddisk.

Register: liten, rask lagringsenhet inne i CPU-en, som brukes til å holde midlertidige data og instruksjoner. Brukes til å holde data som prosessoren trenger umiddelbart under behandling. Flyktig minne.

Cache: hurtigminne som reduserer gjennomsnittlig tilgangstid til data fra hovedminnet. Brukes til å lagre ofte brukte data og instruksjoner for raskere tilgang.

RAM: ligger utenfor CPU'en. Buffer mellom ekstern lagringsmedium og CPU'en med rask lese- og skriveaksess.



SSD/Flash/Harddisk: Høykapasitetsmedium for program/data. «Langt» unna CPU

Datamaskinarkitektur

ALU er den delen av CPU hvor logiske og aritmetiske beregninger utføres

Assemblerkode tolkes til et bitmønster, som har 64 0'ere eller 1'ere, som datamaskinen forstår. De ulike delene av remsen har en mening, og dette må dekodes og det gjøres av CPU. Forsinkelse i CPU skapes av at klokken til CPU'en må ta hensyn til den tregeste prosessen. Problem: hvordan effektivisere? Løsning: Pipeline.

Pipeline fungerer etter samlebåndsprinsippet, ulike prosesser kan skje samtidig. Tenk på Omid som vasker klær. Hvis man har flere subinstruksjoner som skal skje samtidig, vi kan utføre subinstruksjoner fra forskjellige instruksjoner samtidig dersom instruksjonene er ulike. Dette gjør at vi sparer tid, og at det blir mer effektivt. Det at man har en 4 trinns pipeline vil ikke si at man får 4 ganger raskere prosessering, da det alltid vil gå noe tid til administrering av instruksjoner.

- Må lagre verdiene/dataene til hver subinstruks, eller så kan dataen bli overskredet av neste instruks og vil skape trøbbel
- Klokkesignal styrer synkroniseringen

Det kan oppstå tre hazarder ved pipelining:

- **Resource hazard** oppstår hvis to subinstruksjoner i pipelinen ønsker å aksessere samme ressurs
    - Løsninger:
    - Design hvor dette ikke skjer
    - Stoppe (STALL) pipelinen lenge nok til å aksessere sekvensielt
    - Bruke lojale register som er organisert i en registerfil
    - Bruke Harvard arkitekturen som har to separate minner for data og instruksjon
    - (Ingen er særlig gode løsninger)

- **Data hazard** oppstår hvis to forskjellige instruksjoner trenger å aksessere samme data samtidig, den ene instruksjonen vil da trenge et resultat fra forrige instruksjon før den har produsert gyldig svar
    - God løsning:
    - Ved hjelp av forwarding kan man lage en snarvei i pipelinen, i dette tilfellet en direkte datapath som aktiveres ved en hazard
- **Control hazard**: Hvis vi har en pipeline som innhenter neste instruksjon fortløpende vil det oppstå problemer når vi får en JUMP instruksjon (js, jns osv.). Da må vi tømme pipelinen for andre instruksjoner og lese inn den nye
    - Løsning:
    - Ikke hente noe annet før man vet om man skal hoppe eller ikke, forutsi om man skal hoppe, hvis vi dobler hardware så vil vi ha to parallelle løp hvor den ene ikke hopper og andre hopper og deretter bestemmes det til slutt hvilken som skal brukes

Dette er koblingen mellom assemblerkode og maskinkode

KORREKTE PÅSTANDER FRA UNIVERSET

Komponenter

Funksjonaliteten i mikrokjerner implementeres i prosesser som kjører i uberspace

Sekundærminne er nødvendig for å lagre data permanent

Registre er de raskeste lagringsenhetene i en datamaskin

2-er komplement representasjon kan bli brukt for både negative og positive heltall

Vanligvis kjører programmer i brukermodus/ user mode

Transistorer er laget av halvledermaterialer som silisium

Bredden på en BUS (antall linjer) begrenser hvor mye data som kan overføres samtidig

I pipelining kan en instruksjon starte utførelsen før forrige instruksjon er ferdig med dekoding

L1 cache har større lagringskapasitet enn registre

En databuss (BUS) frakter med seg informasjon mellom prosessoren og andre enheter

Styringsenheter (CU) henter instruksjoner fra minnet, dekoder dem og koordinerer utførelsen av operasjonene i prosessoren

Registre er midlertidige lagringsområder inne i prosessoren som holder på data som behandles akkurat nå

ALU er små, raske lagringsenheter inne i prosessoren som holder på data som prosessoren jobber med

RAM lagrer midlertidig programmer og sata som er i aktivt bruk

Sekundærlagring lagrer bilder, videoer og andre filer på langsikt

Synkronisering

En 32 bits binært tall kan representere 2^{32} ulike verdier

Manglende minnekoherens kan føre til feil resultater i beregninger

Pipeline benytter seg av klokkesignal

Synkronisering kan forhindre kappløp (race conditions) mellom prosesser eller tråder

Pipelining i en prosessor kan øke gjennomstrømmingen av instruksjoner

Klokkehastigheten til en prosessor måles vanligvis i gigahertz (GHz)

Synkronisering kan føre til ytelsesproblemer hvis den brukes for mye eller feil

En CPU kan bare utføre aritmetiske operasjoner på tall lagret i registre

Synkronisering er viktig for å koordinere aktiviteter mellom ulike komponenter i et datasystem

Minnekoherens sikrer at alle prosessorer i et system ser den samme verdien av en dataenhet

Registre er den raskeste typen hukommelse i en datamaskin

Minnekoherens er viktig for å sikre at endringer gjort av en prosessor blir synlige for andre prosessorer

Minnekoherens er viktig for å forhindre sikkerhetsproblemer

Kretser

En full-adder kan brukes som subtraktor ved å ha mente-inn = 1

En OR-port med tre innganger gir sann når minst én inngang er sann

En halv-adder har to innganger og gir ut en sum og en mente ut

En AND-port med fire innganger gir bare sann hvis alle inngangene er sanne

En NOR-port med to innganger gir sann hvis og bare hvis begge innganger er falske

En sannhetsverditabell kan ha flere riktige funksjonsuttrykk

Man kan ha fire innganger på en NAND-port

FORKORTELSER

ALU: Aritmetic Logic Unit

CPU: Central Processing Unit

SSD: Solide State-Disker

RAM: Random Access Memory

ISA: Instruction Set Architecture

PC: Program Counter

CU: Control Unit

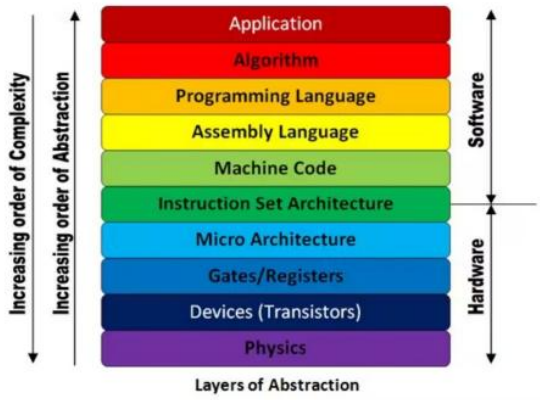
OS: Operating System

GPU: Graphics Processing Unit

USB: Universal Serial Bus

BIOS: Basic Input Output System

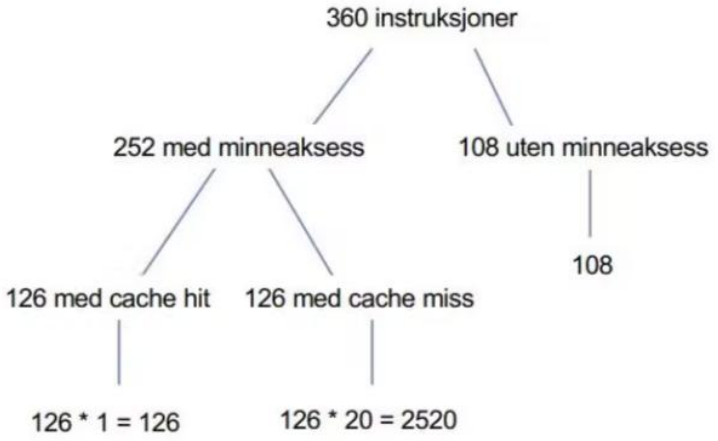
TABELLER / BILDER



Abstraksjonsnivåer for datamaskin

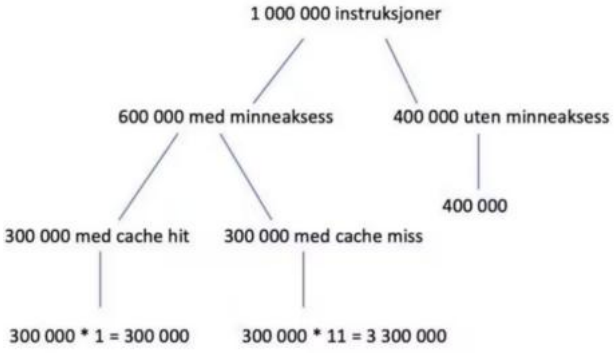
Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\overline{A} = 0$	$A + \overline{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$

Anta at prosessoren har 360 instruksjoner igjen å utføre. Det tar én klokkesykel pr instruksjon, og man regner med å ha en minneaksessering på 70% og med cache-miss på 50%. En cache-miss fører til en total tidsbruk på 20 klokkesykler. Cache-hit bruker totalt én kloykkesykel. Hva er totalt antall gjenstående klokkesykler?



Totalt gjenstående klokkesykler = 126+2520+108 = **2754**

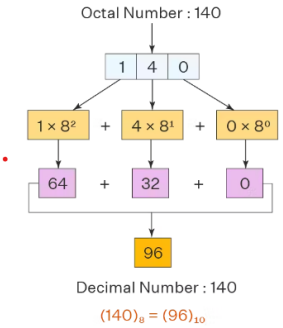
Anta at en prosessor vil prosessere en instruksjon per klokkesykel og skal totalt prosessere 1.000.000 instruksjoner som typisk vil ha behov for å hente data fra minne for 60% av instruksjonene, hvor det vil uheldigvis være 50% cache-miss. Anta videre at det typisk vil være en penalty (økt forsinkelse) for cache-miss på 10 klokkesykler. Hva mange klokkesykler vil det hele ta?



Totalt antall klokkesykler = 3 300 000 + 300 000 + 400 000 = **4 000 000**

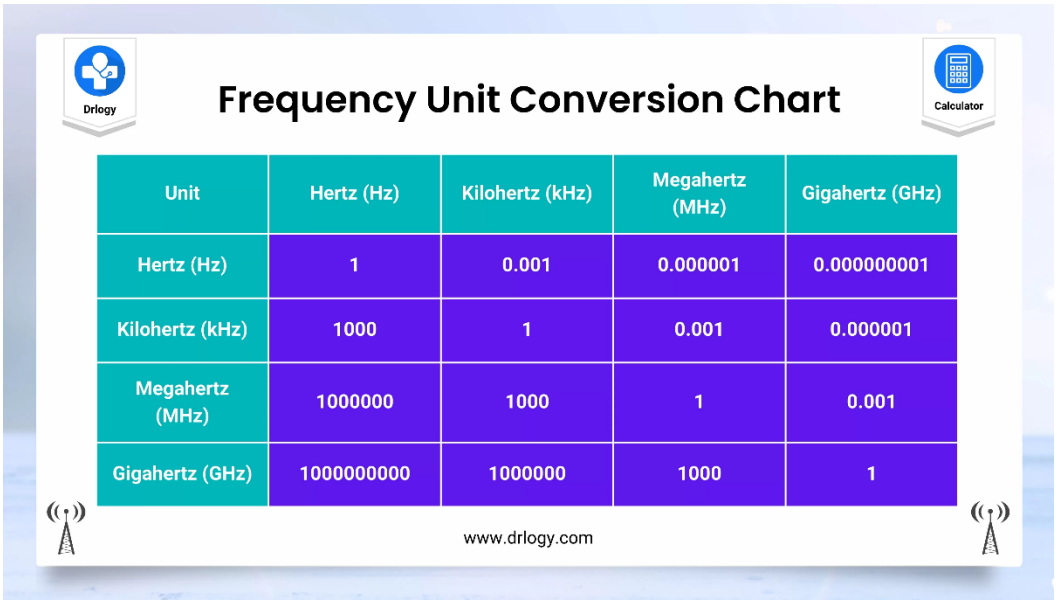
Fra et vilkårlig tallsystem til desimal

- Finn indeks til hvert siffer
- Opphøy basen i indeksen
- Gang med sifferet
- Pluss alle tallene sammen



Memory unit	Description
Kilo Byte	1 KB = 1024 Bytes
Mega Byte	1 MB = 1024 KB
Giga Byte	1 GB = 1024 MB
Tera Byte	1 TB = 1024 GB

METRIC PREFIXES		
Multiple	Prefix	Symbol
10^{-18}	atto	a
10^{-15}	femto	f
10^{-12}	pico	p
10^{-9}	nano	n
10^{-6}	micro	μ
10^{-3}	milli	m
10^{-2}	centi	c
10^{-1}	deci	d
10^{1}	deka	da
10^{2}	hecto	h
10^{3}	kilo	k
10^{6}	mega	M
10^{9}	giga	G
10^{12}	tera	T
10^{15}	peta	P
10^{18}	exa	E



PRØVEEKSAMEN

Abstraksjonsnivå

Høyest til lavest: Pipeline, Register, 2 invertere, logiske verdier, volt

Godt og blandet

Hvilket av disse utsagnene er sanne:

- Det er kun en utgang på en port
- ALU står for Adder-Logikk-Unit
- En XNOR er det inverterte av en XOR
- En utgang av en port kan være inngang på flere porter
- Det er ikke mulig å ha et oddeantall trinn i en pipeline
- Inngangen og utgangen på en inverter er fysisk koblet sammen.
- Man kan lage enhver funksjon med å bare ha INV og OR
- Det finnes bare en type transistorer.

Datapath

Det er **PC** (program counter) som har adressen til neste instruksjon. Nå værende instruksjon blir i dekode-fasen **splittet opp**. Styresignalet til **ALU** kommer fra **DE-steget**. I WB-steget sendes resultatet via en **BUS**.

ALU

Gitt en ALU som skal gjøre følgende operasjoner: Addere, NAND, XOR, NOR, XNOR. Hvor mange bit må styresignalet være? **3 bit** (se styresignal).