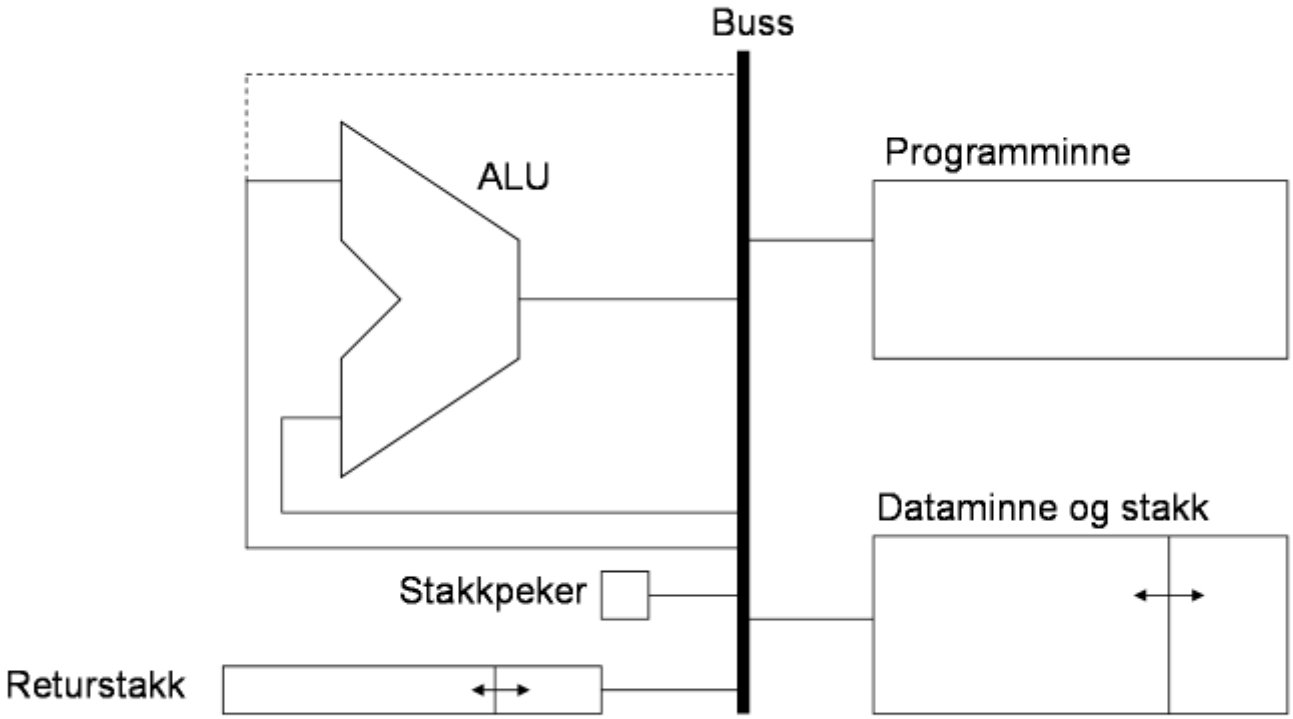
Dark Stakkmaskin



Figur 1: Stakk arkitektur i Dark

Dette dokumentet beskriver arkitekturen til stakkmaskina som benyttes i Dark. Figur 1 viser hvordan maskinen ser ut. Det finnes et register i prosessoren, stakkpekeren (sp). Samtlige beregninger bruker det som ligger på stakken og svaret havner alltid på stakken.

Dette dokumentet inneholder følgende deler:

- Aritmetiske instruksjoner
- Instruksjoner som kontrollerer programflyt
- Instruksjoner som endrer stakken
- Variabeldefinisjon

Aritmetiske instruksjoner

add

Syntaks:

`add`
Semantikk:
stakk ← stakk + stakk
De to øverste verdiene på stakken adresseres og summen av dem blir lagt på stakken. Verdiene som ble addert fjernes fra stakken.

and

Syntaks:
`and`
Semantikk:
stakk ← stakk ∧ stakk
Logisk AND mellom de to øverste verdiene på stakken beregnes og legges på stakken.

band

Syntaks:
`band`
Semantikk:
stakk ← stakk ∧ stakk
Bitvis logisk AND av de øverste to verdiene på stakken. Resultatet blir lagt på stakken.

bnot

Syntaks:
`bnot`
Semantikk:
stakk ← ¬ stakk
Bitvis logiske NOT av den øverste verdien på stakken beregnes og legges på stakken.

bor

Syntaks:
`bor`
Semantikk:
stakk ← stakk ∨ stakk
Bitvis logisk OR av de øvreste verdiene på stakken beregnes. Resultatet legges på stakken.

bxor

Syntaks:
`bxor`
Semantikk:
stakk ← stakk ⊕ stakk
Bitvis logisk XOR av de to øverste verdiene på stakken beregnes og legges på stakken.

div

Syntaks:
`div`
Semantikk:
temp ← stakk
stakk ← stakk / temp
De to øverste verdiene på stakken deles. Resultatet legges på stakken i steden for de to verdiene.

mod

Syntaks:
`mod`
Semantikk:
temp ← stakk
stakk ← stakk % temp
De to øverste verdiene på stakken deles og resten av resultatet legges på stakken.

mul

Syntaks:
`mul`
Semantikk:
stakk ← stakk ∗ stakk
De to øverste verdiene på stakken multipliseres og resultatet blir lagt på stakken istedefor de to verdiene.

not

Syntaks:
`not`
Semantikk:
stakk ← ¬ stakk
Den logiske NOT av den øverste verdien i stakken beregnes og legges på stakken.

or

Syntaks:
`or`
Semantikk:
stakk ← stakk ∨ stakk
Logisk OR mellom de to øverste verdiene på stakken. Resultatet legges på stakken.

sub

Syntaks:
`sub`
Semantikk:
temp ← stakk
stakk ← stakk - temp

De to øverste verdiene på stakken blir subtrahert og resultatet legges på stakken i stedet for de to verdiene.

xor

Syntaks:
`xor`
Semantikk:
stakk ← stakk ⊕ stakk
Logisk XOR av de øverste to verdiene på stakken beregnes og legges på stakken.

Instruksjoner som kontrollerer programflyt

call

Syntaks
`call` ETIKETT
Semantikk
returstakk ← pc
pc ← ETIKETT
Legger adressen til nåværende instruksjon på retrustakken og hopper til neste instruksjon som ETIKETT peker på.

end

Syntaks
`end` {| ETIKETT}
Semantikk
pc ← ETIKETT
Slutt på kildekodefilen. Om ETIKETT er gitt skal prosessoren begynne å eksekvere på denne instruksjonen, ellers starter prosessoren med den absolutt første instruksjonen.

eq

Syntaks:
`eq`
Semantikk:
stakk ← stakk = stakk
Om de to øverste verdiene på stakken er like legges det en 1-er på stakken, ellers legges en 0-er.

ge

Syntaks:
`ge` ETIKETT
Semantikk:
temp ← stakk
stakk ← stakk ≥ temp

Om den nest øverste verdien er lik eller større enn den øverste så legger man en 1-er på stakken, ellers legges en 0-er.

gt

Syntaks:
`gt`
Semantikk:
temp ← stakk
stakk ← stakk < temp
Om den nest øverste verdien er større enn den øverste verdien legges en 1-er på stakken, ellers legges en 0-er.

jfalse

Syntaks:
`jfalse` ETIKETT
Semantikk:
temp ← stakk
{|pc ← ETIKETT}
Om den øverste verdien på stakken er 0 fortsetter eksekveringen med instruksjonen med ETIKETT, ellers skjer ingenting.

jmp

Syntaks:
`jmp` ETIKETT
Semantikk:
pc ← ETIKETT
Eksekveringen fortsetter med instruksjoen med ETIKETT

jtrue

Syntaks:
`jtrue` ETIKETT
Semantikk:
temp ← stakk
{|pc ← ETIKETT}
Om den øverste verdien på stakken er ulik null forsetter eksekveringen med instruksjoenen via ETIKETT, eller skjer ingenting.

le

Syntaks:
`le`
Semantikk:
temp ← stakk
stakk ← stakk ≤ temp
Om den nest øverste verdien på stakken er minder eller lik den øverste verdien en 1-er på stakken, ellers en 0-er.

lt

Syntaks:
`lt`
Semantikk:
temp ← stakk
stakk ← stakk > temp
Om den nest øverste verdien er mindre en den øverste så legges en 1-er på stakken, ellers legges en 0-er.

ne

Syntaks:
`ne`
Semantikk:
stakk ← stakk ≠ stakk
Om de to øverste verdiene på stakken er like legges en 0-er på stakken, ellers legges en 1-er.

nop

Syntaks:
`nop`
Semantikk:
Denne operasjonen gjør ingen ting.

ret

Syntaks:
`ret`
Semantikk:
pc ← returstakk
Eksekveringen hopper tilbake fra en subrutine.

stop

Syntaks:
`stop`
Semantikk:
-
Eksekveringen avsluttes

Instruksjoner som endrer stakken

drop

Syntaks:

`drop`
Semantikk:
? ← stakk
Den øverste verdien på stakken tas bort.

dup

Syntaks:
`dup`
Semantikk:
temp ← stakk
stakk ← temp
stakk ← temp
Den øverste verdien på stakken dupliseres.

pop

Syntaks:
`pop` VARIABEL
Semantikk:
VARIABEL ← stakk
Den øverste verdien på stakken legges i VARIABEL

popa

Syntaks:
`popa`
Semantikk:
temp ← stakk
mem[temp] ← stakk
Den øverste verdien på stakken tolkes som en adresse der neste verdi skal lagres.

pull

Syntaks:
`pull`
Semantikk:
stakk ← mem[stakk]
Den øverste verdien på stakken tolkes som en adresse. Fra denne adressen hentes en verdi og verdien blir lagt på stakken.

push

Syntaks:
`push` { NUMMER| VARIABEL}
Semantikk:
stakk ← { NUMMER| VARIABEL}
Verdien til variabelen/kostnaden legges på stakken.

pusha

Syntaks:
`pusha` VARIABEL
Semantikk:
stakk ← & VARIABEL
Adressen til variabelen legges på stakken

rot

Syntaks:
`rot`
Semantikk:
temp1 ← stakk
temp2 ← stakk
temp3 ← stakk
stakk ← temp1
stakk ← temp3
stakk ← temp2
De tre øverste verdiene roteres slik at det som før låg øverst nå ligger nederst.

swap

Syntaks:
`swap`
Semantikk:
temp1 ← stakk
temp2 ← stakk
stakk ← temp1
stakk ← temp2
De to øverste verdiene bytter plass.

Variabeldefinisjon

area

Syntaks:
`area` NUMMER
Utvider minneområdet til den siste deklarerte variabelen med NUMMER posisjoner og kan for eksempel brukes til å lage arrayer. Dette er det imidlertid ikke behov for i dette kurset.

data

Syntaks:
`data` *NUMMER* NAVN
Om et nummer er gitt kommer varibelen ha denne verdien i begynnelsen av eksekveringen av programmet, ellers så er ikke verdien definert.