ISA Instruction Set Architecture (5)

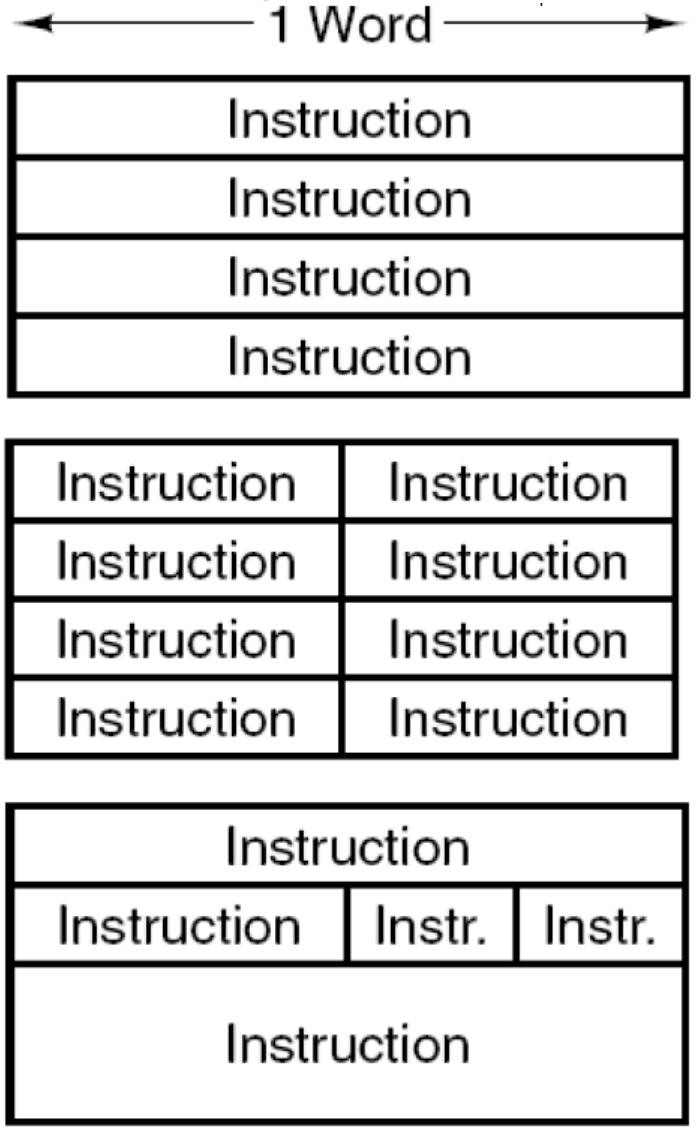
Instruksjonar

- Hvilke instruksjoner som finnes og hvordan de fungerer er helt sentralt i ISA-nivået
- Viktigste typer:
  - Flytting av data
  - Aritmetiske operasjoner
  - Sammenligninger og hoppinstruksjoner
  - Prosedyrekall
  - Løkker
  - I/O

NTNU
Innovation and Creativity

Instruksjonslengde

- Fast instruksjonslengde
  - Enklere dekoding
  - Fordel for samlebånd (spesielt superskalare)
  - Men: mulighet for sløsing med plassen
- Variabel lengde vanlig før, fast lengde vanlig i nye ISA

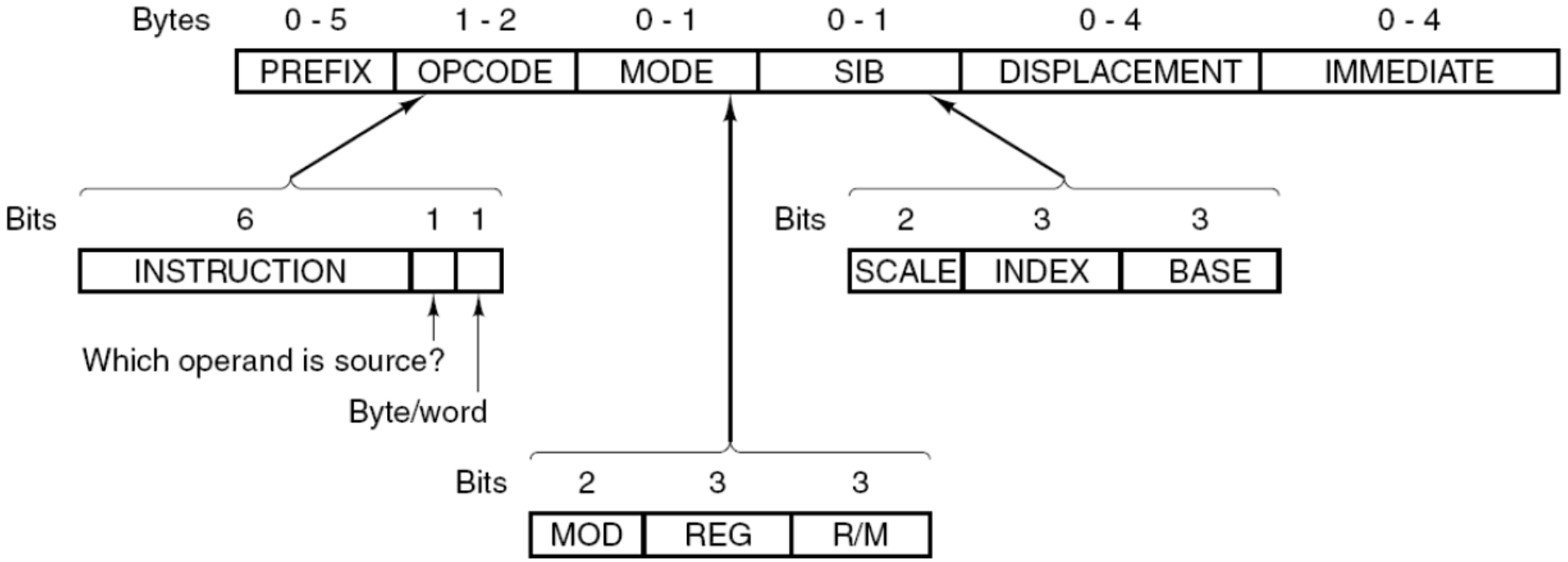


Instruksjonsformat kort/langt

- Opkodefelt
  - Langt felt gir mulighet for mange forskjellige instruksjoner og plass til å vokse på
- # adressefelt – diskutert tidligere
- Adressefelt
  - Langt felt gir mulighet for stort hovedlager, mange registre
- Overordnet
  - Bør være kortest mulig mhp. henting av instr.
  - For kort gir kompleks dekoding, mange begrensninger

P4

- Komplekst og fullt av spesialtilfeller
- 6 felt av variabel lengde, kun 1 er obligatiorisk
- Mye 2-adresseinstruksjoner der 0-1 er hovedlageradr.
- Prefiks lagt til når Opcode ble for lite



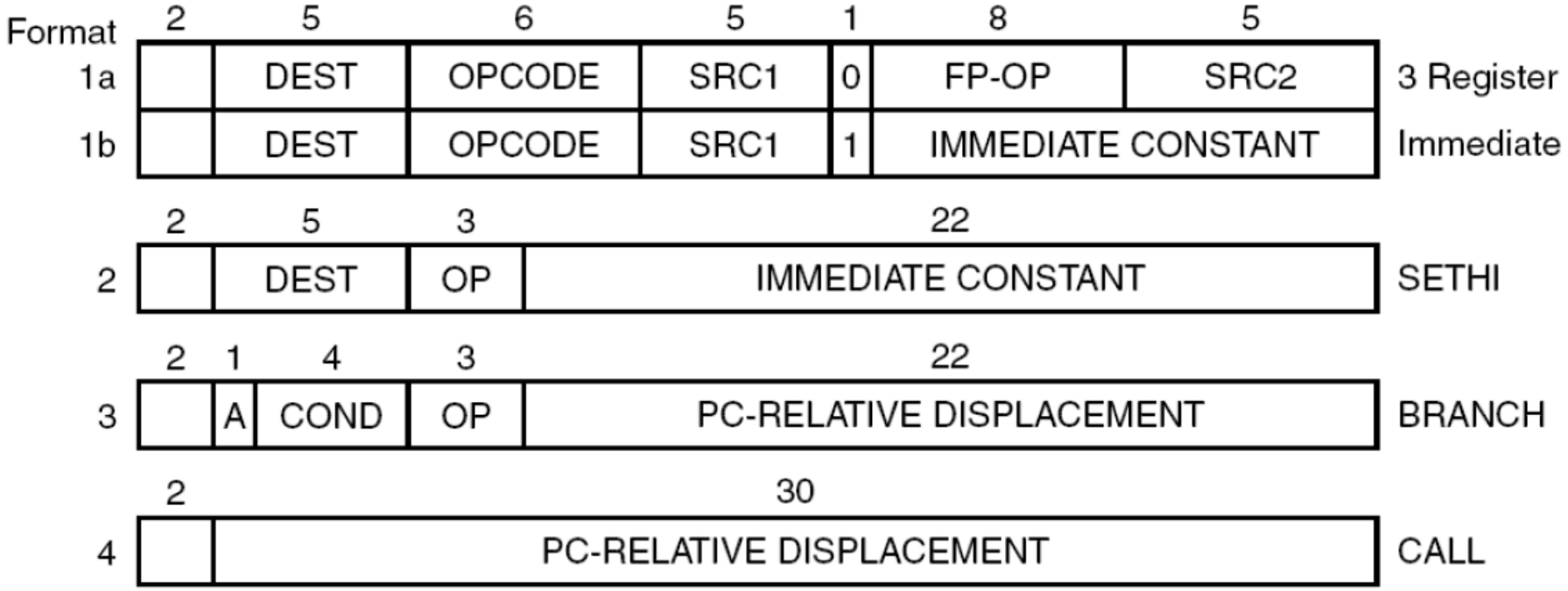
UltraSparc III

- Fast lengde, 32-bits

Som regel 3-adresseinstruksjoner, alt er registre

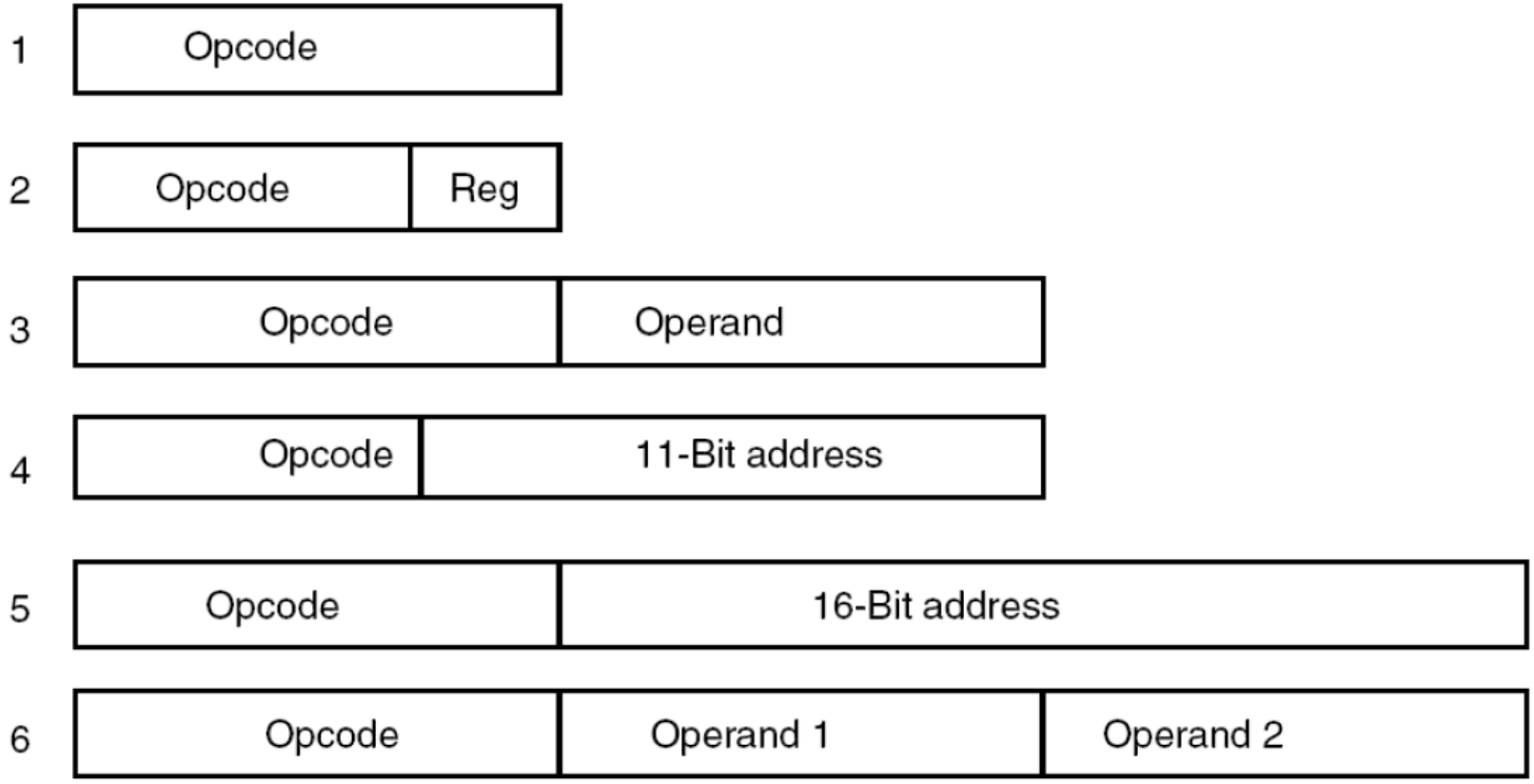
I utgangspunkt kun 4 format, flere lagt til senere

- 2 første bits forteller hvilket av disse 4 det er



8051

- Variabel lengde (plassbruk viktig, ikke samlebånd)
- Bruker akkumulator mye (1-adresseinstruksjoner)

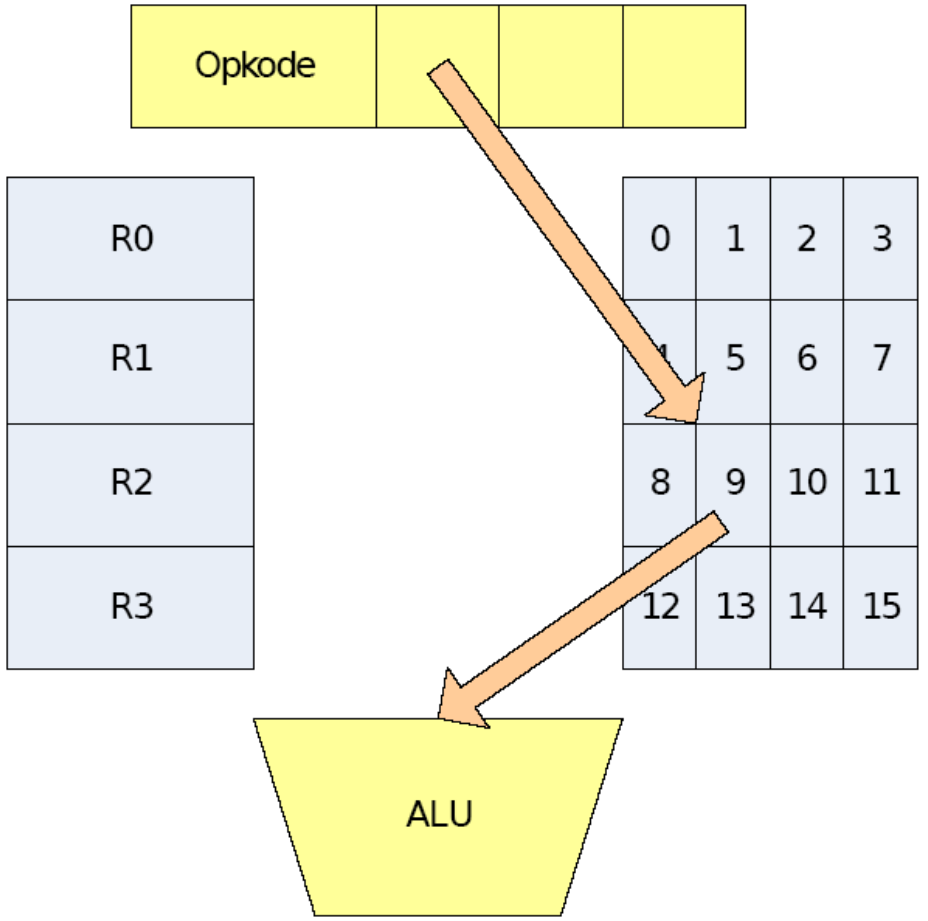


Adresseringsmodi

- Hvordan skal vi oppgi en operand?
- Adresseringsmodus
  - Regel for tolking av adressefelt
  - Mål: Effektiv adresse ➔ Peker på operand
  - Spesifiseres av opkode eller eget modus-felt i instruksjonen
  - Forskjellige operander i samme instruksjon kan ha forskjellig adresseringsmodus

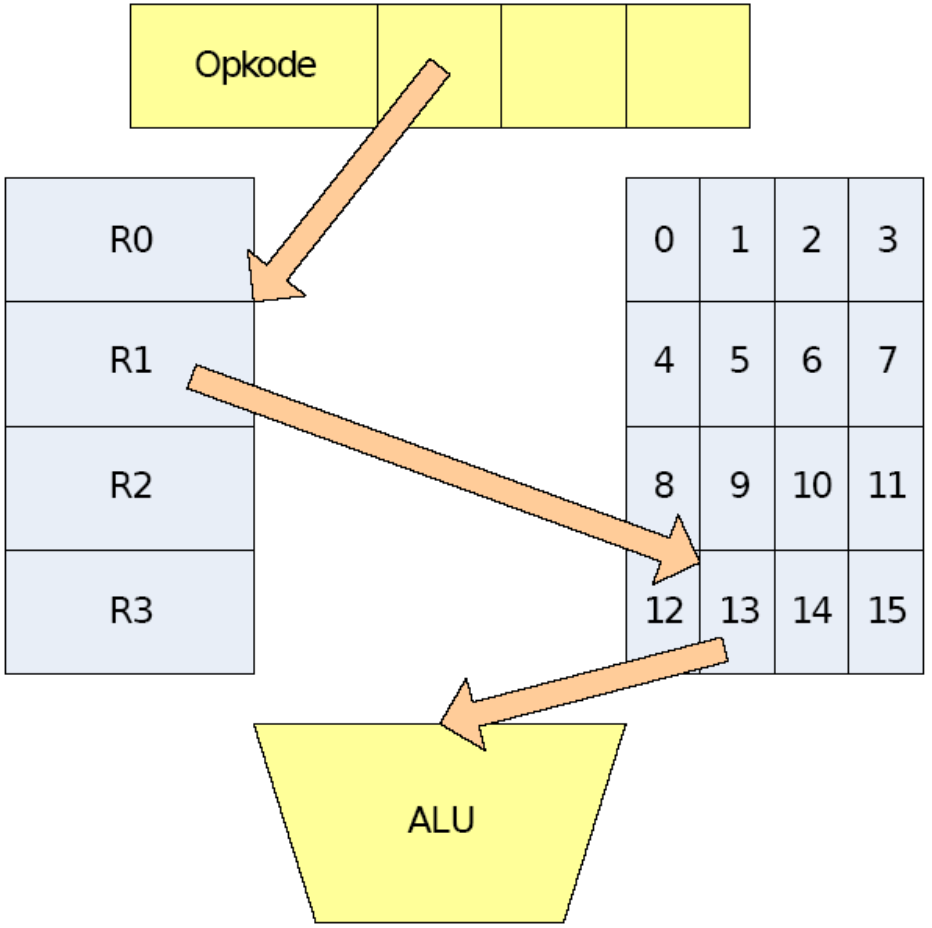
Direkte adressering

- Instruksjonsfelt inneholder hovedlageradresse
- Feltlengde begrenser adresseområde
- Adresse bestemt ved kompilering – lite fleksibelt
- Kan f.eks. bli brukt for globale variable



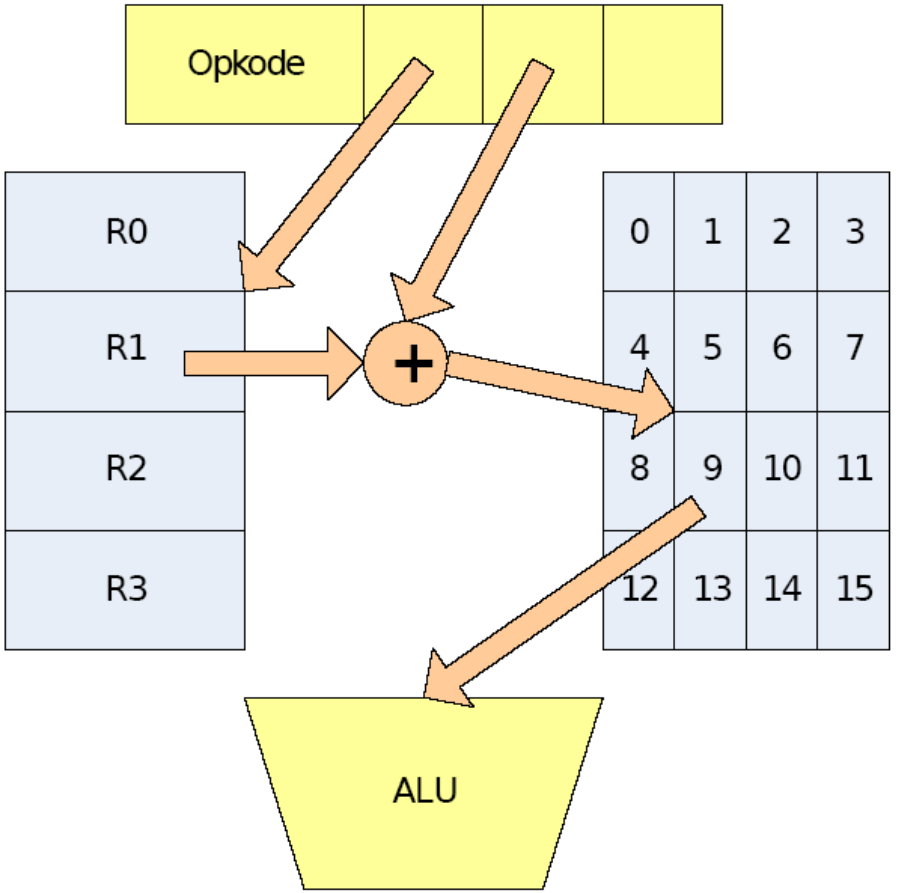
Register-indirekte adressering

- Instruksjonsfelt inneholder registernummer
- Register inneholder hovedlageradresse
  - Kalles *peker*
- Registernummer krever færre bit enn hovedlageradresse
- Fleksibelt



Indeksert adressering

- Instruksjonsfelt inneholder registernummer
- Register inneholder adresse1
- Instruksjonsfelt inneholder adresse2
- adresse1 + adresse2 = hovedlageradresse
- Tilsvarende som for lokale variable i Mic-ark (LV + variabelnummer)

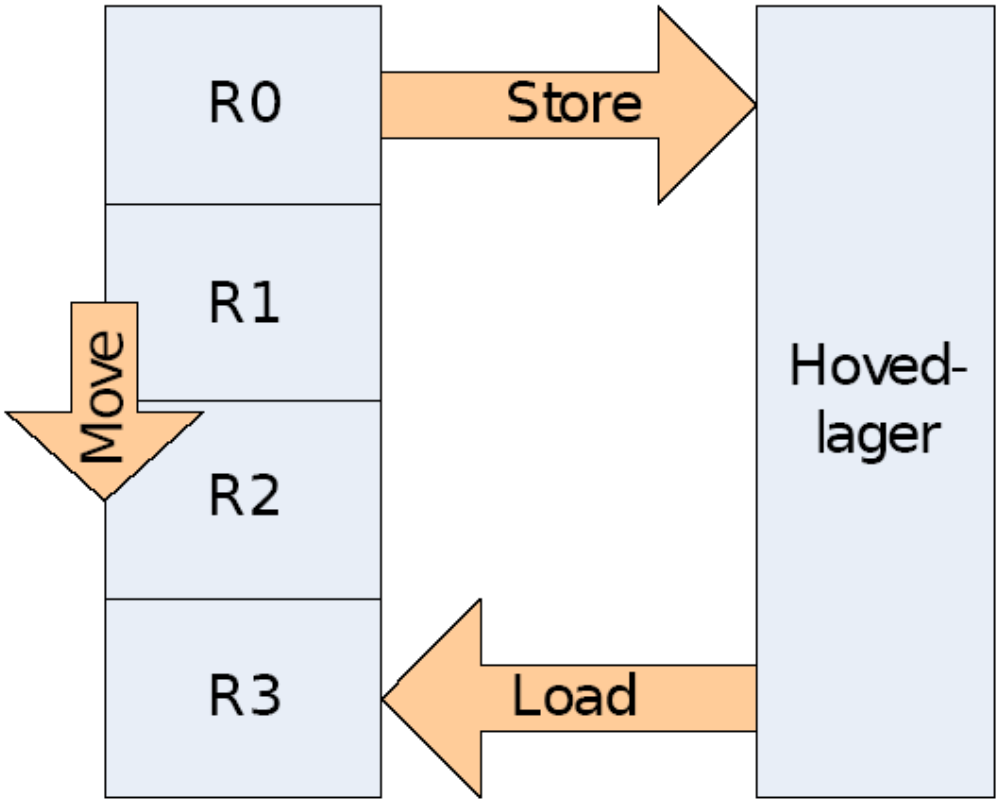


Instruksjonstypar

- Datatransport
- Datamanipulering
- Betingede hoppinstruksjoner
- Prosedyrekall
- Løkker
- I/O

Instruksjonstypar datatransport

- Helt fundamental operasjon
- Snakker gjerne om ”move”, selv om det egentlig er kopiering
- To årsaker
  - Direkte instruksjoner
    Eks: A = B
  - Kopiering av effektivitetshensyn
    Eks: Kopier til register



Instruksjonstypar datamanipulering

- To operander lager ett resultat
- Aritmetiske operasjoner
  - Addisjon, subtraksjon, multiplikasjon, divisjon
  - I alle fall heltall
  - Typisk maskinvarestøtte for flyttall (IEEE 754)
    - Lite effektivt å håndtere i programvare
- Logiske (boolske) operasjoner
  - 16 mulige, men ofte bare ADD, OR (og NOT)

Instruksjonstypar datamanipulering

Eksempler: AND og OR

```
10110111 10111100 11011011 10001011  A
00000000 11111111 00000000 00000000  B (mask)
00000000 10111100 00000000 00000000  A AND B
```

AND kan brukes for å hente ut noen bit av et ord

```
10110111 10111100 11011011 10001011  A
11111111 11111111 11111111 00000000  B (mask)
10110111 10111100 11011011 00000000  A AND B
00000000 00000000 00000000 01010111  C
10110111 10111100 11011011 01010111  (A AND B) OR C
```

OR kan brukes for å legge bit inn i ord

Instruksjonstypar datamanipulering

- En operand lager ett resultat
- Skift
  - Bit som skiftes ut, forsvinner
  - Skift kan utføre mult/div med toerpotenser
  - Konvertering fra parallell til seriell
- Rotering
  - Bit som skiftes ut, kommer inn ”på andre siden”
- Andre: INC, NEG

```
00000000 00000000 00000000 01110011  A
00000000 00000000 00000000 00011100  A shifted right 2 bits
11000000 00000000 00000000 00011100  A rotated right 2 bits
```

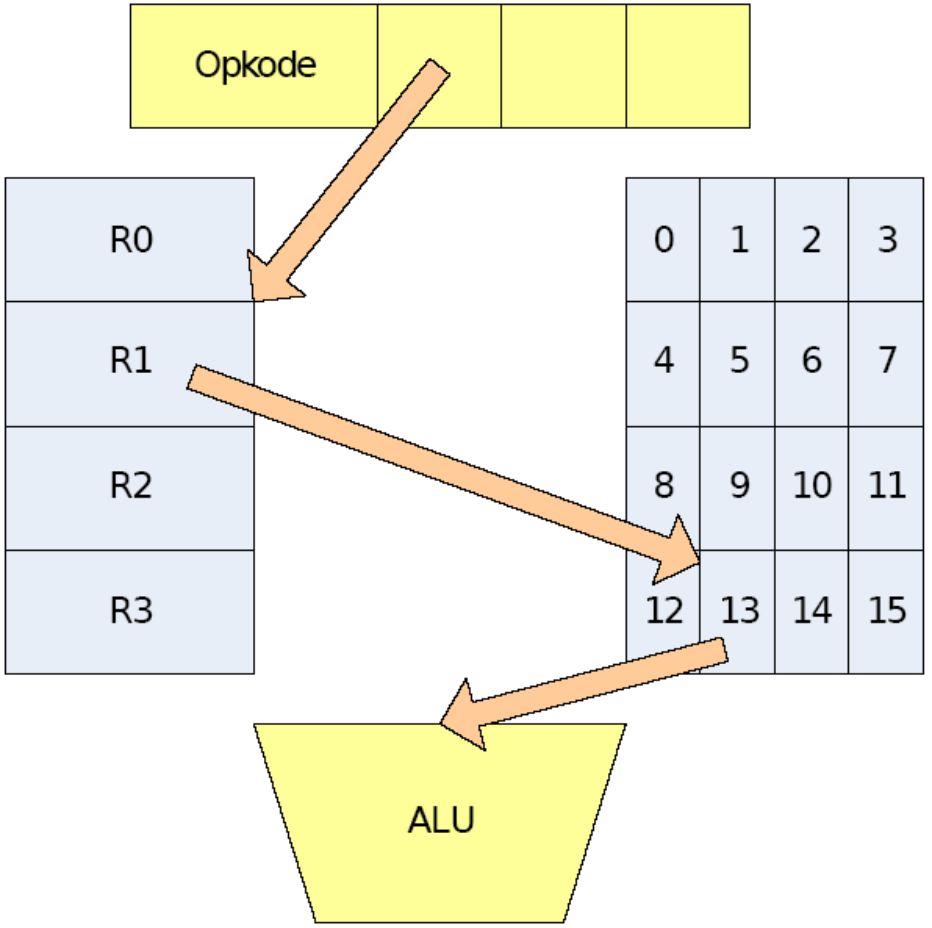
Instruksjonstypar betingahopp

- Hopp til gitt adresse hvis betingelse er tilfredsstilt
  - Har sett hvordan adresse kan oppgis
- Betingelser
  - Operand lik 0, Operand negativ, …
- Vanskelig med betingelse + adresse i en instr.
- Vanlig med to instruksjoner:
  - Sammenligning – setter statusbit
  - Betinget hoppinstruksjon – sjekker statusbit og hopper

```
CMP R0, #0
BEQ LOOP
```

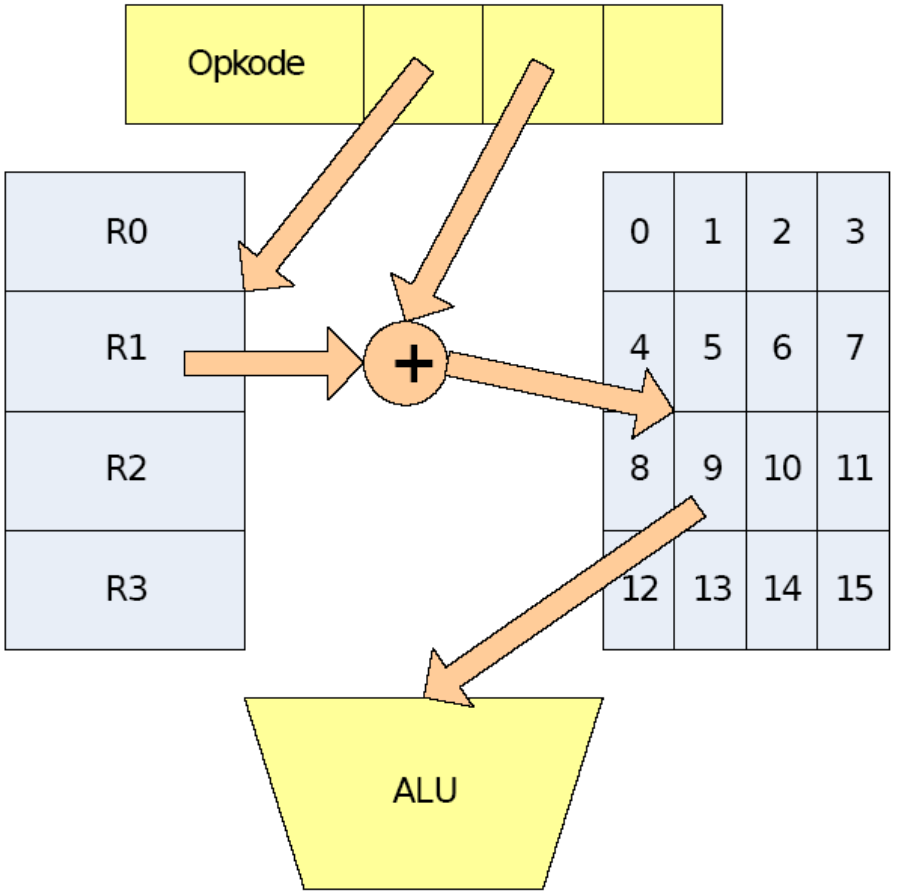
Register-indirekte adressering

- Instruksjonsfelt inneholder registernummer
- Register inneholder hovedlageradresse
  - Kalles *peker*
- Registernummer krever færre bit enn hovedlageradresse
- Fleksibelt



Indeksert adressering

- Instruksjonsfelt inneholder registernummer
- Register inneholder adresse1
- Instruksjonsfelt inneholder adresse2
- adresse1 + adresse2 = hovedlageradresse
- Tilsvarende som for lokale variable i Mic-ark (LV + variabelnummer)



Instruksjonstypar prosedyrekall

- Prosedyre = subrutine ≈ metode (java)
- To steg: Prosedyrekall og returnering
- Trenger derfor å lagre returadresse
  - Hovedlager (dumt)
  - Register (mindre dumt)
  - Stakk (lurt)
- Rekursjon: Se rekursjon

Instruksjonstypar løkker

- Løkker brukes ofte i programkode
  - Kan derfor være lurt med egne instruksjoner
  - Eksempel: En instruksjon som:
    - Trekker 1 fra et register (implisitt el. eksplisitt)
    - Sammenligner resultat med 0
    - Hopper hvis resultat (u)likt 0
- To typer løkker:
  - Test i starten av løkken (java: for, while)
  - Test i slutten av løkken (java: do-while)
    - Dermed blir løkken utført minst en gang

Instruksjonstypar løkker

```
        i = 1;
L1:     first-statement;
        .
        .
        .
        last-statement;
        i = i + 1;
        if (i < n) goto L1;
```

```
        i = 1;
L1:     if (i > n) goto L2;
        first-statement;
        .
        .
        .
        last-statement
        i = i + 1;
        goto L1;
L2:
```

- Ser at test i starten gir mer overhead
  - Likevel dumt å kreve at alle løkker bare skal ha test i slutten

Instruksjonstypar I/O

- Data overføres typisk mellom hovedlager og I/O-enhet
  - Store variasjoner fra ISA til ISA
- 3 vanlige overføringsmodi
  - Programstyrt
  - Avbruddsinitiert
  - DMA-overføring (”Direct Memory Access”)

Instruksjonstypar datamanipulering

- To operander lager ett resultat
- Aritmetiske operasjoner
  - Addisjon, subtraksjon, multiplikasjon, divisjon
  - I alle fall heltall
  - Typisk maskinvarestøtte for flyttall (IEEE 754)
    - Lite effektivt å håndtere i programvare
- Logiske (boolske) operasjoner
  - 16 mulige, men ofte bare ADD, OR (og NOT)

Instruksjonstypar I/O avbrudd

- Programstyrt I/O
  - Prosessor sjekker om data er klart
- Avbruddsinitiert I/O
  - I/O-enhet sier ifra når data er klart
  - Unngår å kaste bort prosessortid
- Adresse til avbruddsrutine
  - Ikke-vektorisert: Fast, en for alle avbrudd
  - Vektorisert: Avbruddskilde oppgir adresse
- Mer om avbrudd senere

Instruksjonstypar DMA

- Mye dataoverføring gir prosessor lite tid til annet
- Kan ikke andre ta jobben?
- Direct Memory Access
  - Overføring direkte til/fra hovedlager
  - Prosessor tar seg kun av oppstart av overføring
    - Enhet, startadresse, datalengde
- Prosessor kan dermed gjøre annet imens
  - Men: Ikke aksessere hovedlager!
  - DMA høyere prioritet enn CPU, kan ofte ikke vente
  - DMA ”stjeler” buss-sykler fra CPU – ”cycle stealing”

Flytkontrol (flow control)

- Normalt: Sekvensiell utføring
- Flytkontroll: Dynamisk endring av instruksjonsrekkefølge under utføring
- Typer
  - (Betinget) Hopp
  - Prosedyrekall (Java: Metodekall)
  - Ko-rutiner
  - Trap / Avbrudd (”Interrupt”)

NTNU
Innovation and Creativity

Flytkontrol (flow control)