FKEF.02.094

Arvuti arhitektuur

Computer Architecture and Organization

dotsent
Toomas Plank

FKEF.02.094

Masinkood (1)

3. loeng,
7. märts 2008

Jutujuht

Selles loengus tutvume:

- mälu adresseerimisega
- masinkoodiga ja programmide ülesehitusega
  - Sirgjooneline käsutäitmine
  - Hargnemine
  - Alamprogrammide väljakutsumine
  - Adresseerimine
- Järgmises loengus
  - Sisend/väljund operatsioonid
  - Pinu, järjekord, andmetabel
  - Nihe, pööre masinkoodis

FKEF.02.094

Mälu adresseerimine

Mälu

- Mälu koosneb miljonitest rakkudest
- Igas rakus saame salvestada ühe biti infot
  - ehk siis võimalikud olekud on 0 ja 1
- Tavaliselt loetakse-kirjutatakse infot *n* biti kaupa
- andmeühikuks on sõna, ingl *word*, kus *n* on sõna pikkus
- Tänapäevastes arvutites on sõna pikkuseks 16 – 64 bitti
- 8-bitist andmeühikut kutsutakse baidiks (*byte*)

Mälu on sõnade kogum

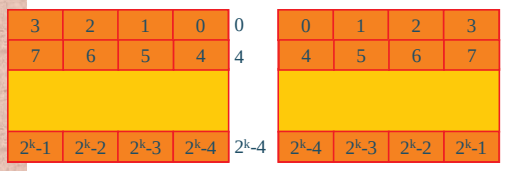
- Igal mälupesal on oma kindel nimi – aadress
- Tavaliselt on selleks numbrid 0 … 2^k-1
- Seega 2^k võimalikku adresseeritavat asukohta
- Näiteks 2^{24} = 16 Mi = 16 777 216
- Näiteks 2^{32} = 4 Gi = 4 294 967 296

Baidi aadressid sõnas

Big-endian
Väiksema aadressiga mälupesad sisaldavad arvu kõrgemaid baite

- *byte-addressable memory*
- s.t 32-bitiste sõnade korral algavad sõnad aadressidelt 0, 4, 8 …
- On kaks varianti sõna sees baitide paigutamiseks

Baidi aadressid sõnas



Little-endian
Väiksema aadressiga mälupesad sisaldavad arvu madalamaid baite

Big-endian
Väiksema aadressiga mälupesad sisaldavad arvu kõrgemaid baite

Sõna

- Selline nummerdamine on kõige loomulikum viis numbrite kirjapanemiseks
- Sama skeemi kasutatakse ka baidi sees bittide nummerdamiseks

Sõnade paiknemine

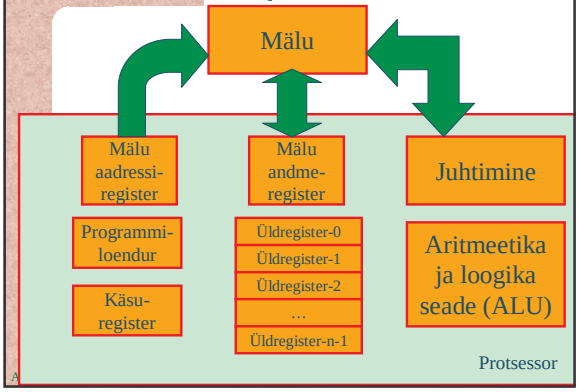
- Kaks võimalust
  - *aligned adresses*

  ja
  - *unaligned adresses*
- Esimesel juhul algavad sõnad kohtadel, mis on sõna baitide arvu kordsed, teisel juhul mitte
- Numbrid võtavad enda alla tavaliselt ühe sõna, nende poole pöördumiseks kasutatakse sõna aadressi
- Tähed – pöördumisel kasutame baidi aadressi
- Tekst – viidatakse algustähe aadressile ja lõpp tähistatakse spetsiaalse sümboliga *End-of-string*
  - Alternatiiviks on teksti pikkuse näitamine spetsiaalses registris

Mälu operatsioonid

- Programmi käsud tuleb tuua mälust protsessorisse
- Parameetrid ja andmed tuleb samuti tuua mälust protsessorisse
- Vastus tuleb salvestada protsessorist mällu
- Load, Read, Fetch
  - Ütleme, mis aadressil paikneb meid huvitav info
  - Loeme sealt aadressil info protsessorisse
- Store, Write
  - Ütleme, millisesse mälupesasse tahame andmeid kirjutada
  - Kirjutame andmed viidatud aadressiga mälupesasse

Tööpõhimõte



FKEF.02.094

Masinkoodi käsud ja käskude järjekord

Tänase loengu materjali asukoht kihtstruktuuris

- Arvuti võib jagada seitsmeks üksteisest suhteliselt sõltumatuks tasemeks:

Kasutaja tase: rakendusprogrammid
Kõrgtaseme keeled
Assembly language / Masinkood
Juhtimise tase
Funktsionaalsed elemendid
Loogikaahelad
Transistorid ja traadid

Arvuti ülesanded

- Andmete liigutamine protsessori registrite ja mälu vahel
- Aritmeetika ja loogikaoperatsioonide tegemine andmetega
- Programmi käskude järjestamine ja juhtimine
- Sisend/väljund operatsioonid

Register Transfer Notation

- R0←[MÄLUPESA_1]
- R0←[M1]
- R4←[R0]+[R1]

Assembly Language Notation

- R0←[M1]
  Move M1,R0
- R4←[R0]+[R1]
  Add R0,R1,R4

Põhilised käsu tüübid

- Vastus := muutuja1 + muutuja2;
- M0←[M1]+[M2]
- Ühe masinkäsuga
  - Add M1,M2,M0
  - Lähteandmed M1 ja M2 (mõlemad k-bitised)
  - Tulemus M0 (ka see on k-bitine)
- Tegelikult ei mahu ühte sõnasse ära L
- Kahe aadressiga variant
  - Add M1,M2
  - NB! Veidi erinev eelmisest käsust!
  - M2←[M1]+[M2]

Põhilised käsu tüübid

- Kahe aadressiga kahekäsuline variant
  - Move M2,M0
  - Add M1,M0
  - Nüüd uuesti M0←[M1]+[M2]
- Tegelikult ka need ei mahu alati ühte sõnasse ära L
- Ühe aadressiga variant:
  - Load M2
  - Add M1
  - Store M0
  - Teine argument siin on akumulaator
  - Sõltuvalt käsust kas lähteandmed või tulemus

Registrid

- Andmete ülekandmine mälu ja protsessori registrite vahel
  - Mälu aadressid võtavad käsus palju ruumi
  - Protsessori registrite aadressid on oluliselt lühemad (näiteks 32-bitise registri korral 5 bitti)
  - Registritega suhtlus on ka kiirem kui suhtlus mäluga
- Meie eelmine näide võtab kuju:
  - Load M2,Ri
  - Add M1,Ri
  - Store Ri,M0
- ehk
  - Move M2,Ri
  - Add M1,Ri
  - Move Ri,M0

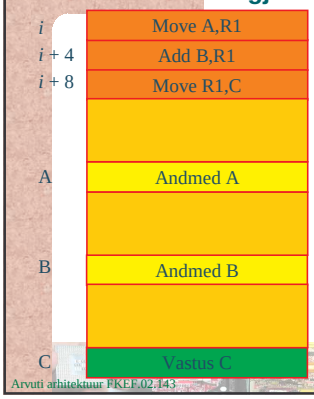
Registrid

- Kui aritmeetikaoperatsioone lubatakse ainult registris olevate andmetega, siis
  - Move M2,Ri
  - Move M1,Rj
  - Add Ri,Rj
  - Move Rj,M0

Käsud ja nende järjestus

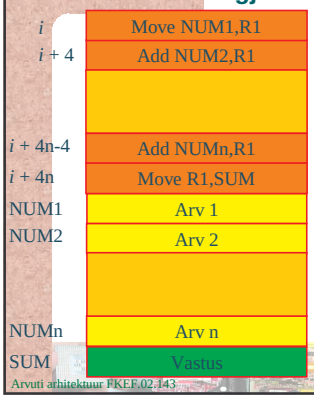
- Sirgjooneline käsutäitmine
- Hargnemine

Sirgjooneline käsutäitmine



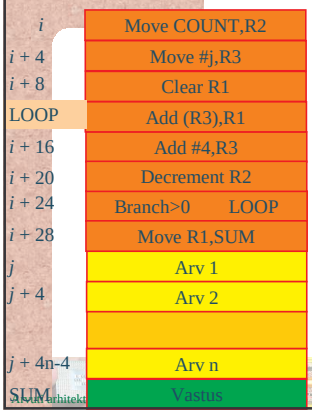
- Programmiloendur (PC) viitab käsule aadressil *i*
- Järgmine käsk on ühe astme (4) võrra kaugemal – s.t kohal *i* + 4
- Sirgjoonelise järjestuse korral peame iga käsu täitmise ajal suurendama PC ühe astme võrra
- Käsud on kaheastmelised:
  - *instruction fetch*
  - *instruction execute*

Sirgjooneline käsutäitmine (2)



- Oletame, et meil on vaja liita palju numbreid
- Ilmselt on sirgjoonelise lahenduse programmeerimine üsnagi aegavõttev…

Hargnemine



- Kasutame ühte Add käsku ja hargnemist, kus kontrollime, kas on veel mõni arv jäänud liitmata

Staatus-register

- *Condition code register* või *status register*
- Vastavalt tulemusele seatakse lipud kas 0 või 1
  - N (negatiivne) – kui tulemus on negatiivne siis 1; muidu 0
  - Z (null) – kui tulemus on null siis 1; muidu 0
  - V (ületäitumine) – kui leidis aset aritmeetiline ületäitumine siis 1; muidu 0
  - C (ülekanne) – kui leidis aset ülekanne (*carry-out*) siis 1; muidu 0
- Mõnedes protsessorites muutuvad lipud automaatselt, teistes on eraldi käsud, millega lippude muutmine kaasneb
  - Add (ei muuda lippe)
  - AddSetCC (muudab lippe)

Adresseerimisviisid

Ingliskeelne nimi	Assembler süntaks	Aadressi kujunemine
Immediate	#väärtus	operand = väärtus
Register	R_i	EA = R_i
Absolute (Direct)	LOC	EA = LOC
Indirect	(R_i)	EA = $[R_i]$
Index	$X(R_i)$	EA = $[R_i]+X$
Base with index	(R_i, R_j)	EA = $[R_i]+[R_j]$
Base with index and offset	$X(R_i, R_j)$	EA = $[R_i]+[R_j]+X$
Relative	X(PC)	EA = [PC]+X
Autoincrement	$(R_i)+$	EA = $[R_i]$; R_i++
Autodecrement	$-(R_i)$	R_i--; EA = $[R_i]$

Adresseerimise näited (1)

- Registri mood – operandiks on protsessori registri sisu, käsus antakse registri aadress (nimi)
- Absoluutne (otsene) mood – operandiks on mälupesa sisu, käsus antakse selle aadress
- Näide:
  - Käsuga int arv1, arv2; kirjeldame C-keeles täisarvulised muutujad arv1 ja arv2. Neile vastavad konkreetsed mälupesad, millele saab absoluutses moodis viidata
  - Käsus move LOC,R5 on esimene operand esitatud absoluutses moodis, teine aga registri moodis
  - esimese operandi väärtuseks on **mälupesa LOC sisu**, teise operandi väärtuseks **registri R5 sisu**

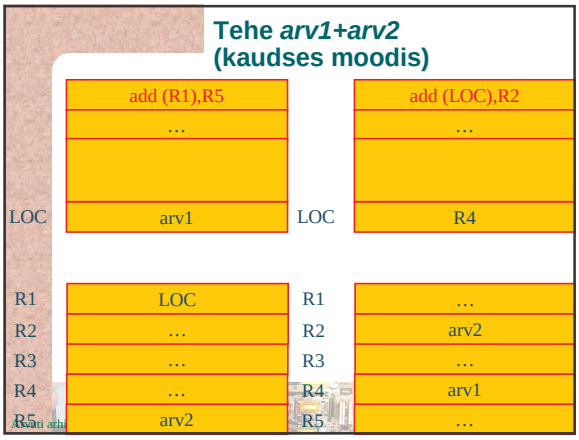
Adresseerimise näited (2)

- *Immediate* mood – operandi väärtus antakse juba käsus endas
- Näide:
  - Käsus move #500,R5 on esimese operandi väärtuseks 500, teise operandi väärtuseks aga registri R5 sisu
  - Näiteks tehte *arv1 = arv2 + 8;* saame kirja panna
    move arv2,R1
    add #8,R1
    move R1,arv1
  - arv1 ja arv2 tuleb loomulikult eelnevalt muutujatena deklareerida ja mälupesaga seostada

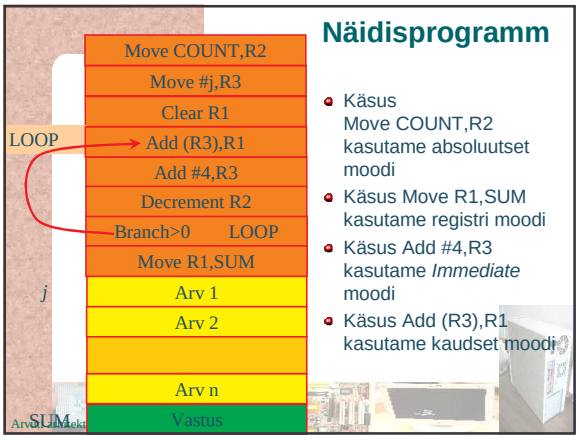
Adresseerimise näited (3)

- Kaudne mood – operandi efektiivne aadress antakse mälu või registri aadressina kus **operandi aadress** kirjas on
- Näide:
  - Käsus add (R1),R5 on esimese operandi väärtuseks **registris R1 viidatud aadressil asuva mälupesa või registri sisu**, teise operandi väärtuseks aga registri R5 sisu
  - Käsus add (LOC),R2 on esimese operandi väärtuseks **mälupesas LOC viidatud aadressil asuva mälupesa või registri sisu**, teise operandi väärtuseks aga registri R2 sisu
  - registrit või mälupesa, kus efektiivne aadress kirjas on kutsutakse *pointer*'iks

Tehe *arv1+arv2* (kaudses moodis)

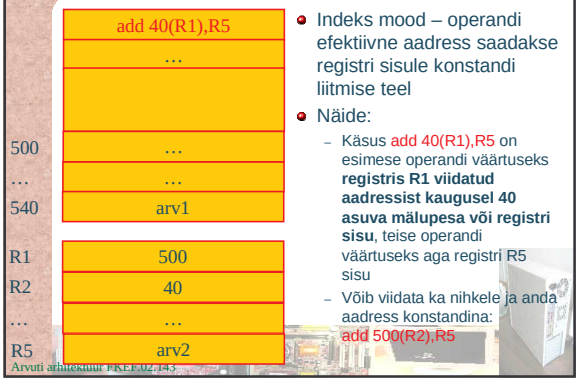


Näidisprogramm



- Käsus Move COUNT,R2 kasutame absoluutset moodi
- Käsus Move R1,SUM kasutame registri moodi
- Käsus Add #4,R3 kasutame *Immediate* moodi
- Käsus Add (R3),R1 kasutame kaudset moodi

Adresseerimise näited (4)



- Indeks mood – operandi efektiivne aadress saadakse registri sisule konstandi liitmise teel
- Näide:
  - Käsus add 40(R1),R5 on esimese operandi väärtuseks **registris R1 viidatud aadressist kaugusel 40 asuva mälupesa või registri sisu**, teise operandi väärtuseks aga registri R5 sisu
  - Võib viidata ka nihkele ja anda aadress konstandina: add 500(R2),R5

Näide

- Kasutatakse andmetabelite puhul
- Olgu meil isikuandmete andmebaas, kus säilitame iga kodaniku kohta tema arveldusarvel oleva rahasumma suurust
  - Hansapangas,
  - SEB's

  ja
  - Sampo pangas
- Kirjutame programmi, mis iga panga lõikes hoiuse väärtused kokku liidaks
  - sisuliselt leiame nii panga turuosa (arveldus)hoiuste turul

Näide

	Move #j,R0	
	Clear R1	
	Clear R2	
	Clear R3	
	Move N,R4	
LOOP	Add 4(R0),R1	
	Add 8(R0),R2	
	Add 12(R0),R3	
	Add #16,R0	
	Decrement R4	
	Branch>0 LOOP	
	Move R1,SUM1	
	Move R2,SUM2	
	Move R3,SUM3	

R0	j + 16
j	Kodaniku 1 ID
j + 4	Konto Hansapangas
j + 8	Konto SEB's
j + 12	Konto Sampo pangas
j + 16	Kodaniku 2 ID
j + 20	Konto Hansapangas
j + 24	Konto SEB's
j + 28	Konto Sampo pangas
SUM1	Vastus 1
SUM2	Vastus 2
SUM3	Vastus 3

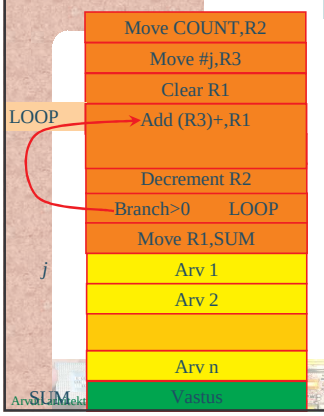
Adresseerimise näited (5)

- Indeks moodiga väga sarnane mood on baas koos indeksiga
  - (R_i, R_j)
  - Viidatakse aadressile $R_i + R_j$
- ja baas koos indeksi ja nihkega
  - X(R_i, R_j)
  - Viidatakse aadressile $X + R_i + R_j$
  - Seda kasutatakse kolmemõõtmelise massiivi kirjeldamisel
- Suhteline mood – sisuliselt indeks mood, aga registriks on PC (*Program Counter*)
  - tüüpiline kasutus Branch > LOOP käsu sihtkoha kirjeldamiseks. Näiteks kujul –16(PC)

Adresseerimise näited (6)

- *Autoincrement* mood -- väga sarnane kaudse moodiga. Operandi efektiivne aadress antakse mälu või registri aadressina kus **operandi aadress** kirjas on. Peale operandi kättesaamist suurendatakse registri sisu ühe **ühiku** võrra.
  - ühik valitakse selline, et aadress viitaks järgmisele elememendile
  - 1 baidisuuruse operandi puhul,
  - 2, kui operand on 16 bitti
  - 4, kui operand on 32 bitti
  - Tähis (Ri)+
- *Autodecrement* mood – **Enne** operandi kättesaamist vähendatakse registri sisu ühe **ühiku** võrra. Operandi efektiivne aadress antakse mälu või registri aadressina kus **operandi aadress** kirjas on.
  - Tähis –(Ri)

Näidisprogramm



- Käsud
  Add (R3),R1
  ja
  Add #4,R3
- saame kokku võtta käsuks Add (R3)+,R1

Kasutatud kirjandus

- Carl Hamacher, Zvonko Vranesic, Safwat Zaky, Computer organization 5th edition (2002) 805 p.