Programski prevodioci 1

Generisanje koda

Generisanje koda

- Ulaz u generator koda je međukod programa, izlaz je mašinski kod za ciljnu mašinu.
- moguće varijante: apsolutizovani mašinski kod, objektni kod obrađivan u predmetu sistemski softver, asemblerski izvorni kod-ako izlaz ide potom u asembler.

Ciljna mašina za ovu lekciju: podskup x86 asemblera

- Dvoadresna mašina, asemblerske naredbe oblika:
  OP destination, source
  ima značenje destination := destination OP source
- Ograničenje: ne mogu istovremeno i source i destination biti memorijske lokacije (jedan mora biti u registru)
- OP će nam biti ADD, MOV, SUB, MUL, DIV.
  - MUL src ima značenje: (DX:AX) ← AX * src
  - DIV src ima značenje: AX ← (DX:AX) div SRC
- DX ← (DX:AX) mod SRC
- Ima registare opšte namene AX, BX, CX, DX i indeksne SI, DI, BP.
- Adresiranje na nivou bajta, memorijska reč od 2 bajta. U našim primerima svi operandi i rezultati operacija će biti 16bitni.

Načini adresiranja

Način adresiranja	Sintaksa	Objašnjenje
Apsolutno	M	Simbolička mem. adresa M
Registarsko	R	Registar R
Indeksno (r.relat)	[R±c]	Konstanta c ± sadržaj reg. R
Indirektno reg.	[R]	Sadržaj registra R
Neposredno	c	Konstanta c

Napomena: kod reg. indirektnog mogu se samo koristiti registri BX, SI, DI. Kod indeksnog samo registri BX, BP, SI, DI.

Funkcije generatora koda

1) **Izbor naredbi** (instruction selection)

- Mapiranje IR u mašinski kod
- Podrazumeva se fiksirano mapiranje promenljivih u memoriju i neograničen broj registara
- Problem izbora operacija, načina adresiranja

2) **Raspoređivanje naredbi** (instruction scheduling)

- Promena redosleda naredbi da se smanje kašnjenja, upotreba registara i slično
- Podrazumeva se fiksirani program, tj. skup instrukcija

3) **Dodela registara** (register allocation)

- Odlučivanje o mapiranju promenljivih u memoriju i registre procesora
- Menja mapiranje promenljivih u memoriju, moguće je (privremeno) deljenje iste lokacije od strane više promenljivih

Ove 3 funkcije su tesno povezane, što usložnjava generisanje koda.

Neki problemi generisanja koda

- neefikasan kod ako se pojedini iskazi posmatraju odvojeno:

Primer

Generisati mašinski kod za naredbe:

a = b + c

d = a + e

Generisani kod:

MOV AX, b

ADD AX, c

MOV a, AX } nepotrebno

MOV AX, a

ADD AX, e

MOV d, AX

Neki problemi generisanja koda

- problem izbora naredbi.

Primer

Izraz a = a +1

može se prevesti na (bar) dva načina:

```
                MOV AX, a
INC a   ili     ADD AX, 1
                MOV a, AX
```

- problem dodele registara promenljivim, odnosno izbora skupa promenljivih koje će biti u registrima u datoj tački izvršavanja programa, što ima neposredan odraz na efikasnost koda, odnosno broj pristupa memoriji. Problem je NP kompletan (rešenje eksponencijalno zavisi od broja promenljivih)

Jednostavan algoritam generisanja koda za jedan bazični blok

- Prema knjizi: Aho, Lam, Sethi, Ullman „Compilers/Principles, Techniques, and Tools”.
- u prvom koraku određuju se informacije o **životu** i **sledećem korišćenju** za svaku pojavu promenljivih u bazičnom bloku.
- Promenljiva x je ***živa*** u iskazu *i* programa ako postoji putanja do nekog iskaza j po kojoj se x ne menja, a u iskazu j promenljiva x se koristi kao operand. Tada iskaz j predstavlja ***sledeće korišćenje*** promenjive x. U suprotnom, za promenljivu x se kaže da je *mrtva*.

Postupak izračunavanja života i sledećeg korišćenja promenljivih

1. Ako se ne vrši globalna analiza, konzervativna pretpostavka je da su na izlazu iz bazičnog bloka sve korisničke promenljive žive, a privremene mrtve (postoje izuzeci). Ove informacije sačuvamo u tabeli simbola (T.S) za svaku promenljivu.
2. Iskazi u bazičnom bloku obrađuju se redosledom od poslednjeg ka prvom. Kada naiđemo na i-tu četvorku
   i. **x := y op z ;**

   2.1. iz T.S. pokupimo informacije o x, y, z i prikačimo i-toj četvorci.

   2.2. u T.S. ažuriramo x = (m , -), to jest, x je mrtvo i nema sledeće korišćenje

   2.3. u T.S. ažuriramo: y, z = (ž , i), to jest, y i z su živi a iskaz i predstavlja sledeće korišćenje

Primer

Generisati mašinski kod za sledeću instrukciju :

d := (a – b) + (a – c) + (a – c).

Primer

Generisati mašinski kod za sledeću instrukciju :

d := (a – b) + (a – c) + (a – c).

Međukod :

1. t := a – b
2. u := a – c
3. v := t + u
4. d := v + u

Primer

Generisati mašinski kod za sledeću instrukciju :

d := (a – b) + (a – c) + (a – c).

Međukod :

1. t := a – b
2. u := a – c
3. v := t + u
4. d := v + u

Tabela simbola

a	b	c	d	t	u	v
(ž , -)	(ž , -)	(ž , -)	(ž , -)	(m , -)	(m , -)	(m , -)

Primer

Generisati mašinski kod za sledeću instrukciju :

d := (a – b) + (a – c) + (a – c).

Međukod :

1. t := a – b
2. u := a – c
3. v := t + u
4. d := v + u **d = (ž , -)** **v = (m , -)** **u = (m , -)**

Tabela simbola

a	b	c	d	t	u	v
(ž , -)	(ž , -)	(ž , -)	(ž , -)	(m , -)	(m , -)	(m , -)

Primer

Generisati mašinski kod za sledeću instrukciju :

d := (a – b) + (a – c) + (a – c).

Međukod :

1. t := a – b
2. u := a – c
3. v := t + u
4. d := v + u d = (ž , -) v = (m , -) u = (m , -)

Tabela simbola

a	b	c	d	t	u	v
(ž , -)	(ž , -)	(ž , -)	~~(ž , -)~~ **(m , -)**	(m , -)	~~(m , -)~~ **(ž , 4)**	~~(m , -)~~ **(ž , 4)**

Primer

Generisati mašinski kod za sledeću instrukciju :

d := (a – b) + (a – c) + (a – c).

Međukod :

1. t := a – b
2. u := a – c
3. v := t + u
4. d := v + u

v = (ž , 4) d = (ž , -)

t = (m , -) v = (m , -)

u = (ž , 4) u = (m , -)

Tabela simbola

a	b	c	d	t	u	v
(ž , -)	(ž , -)	(ž , -)	~~(ž , -)~~ (m , -)	(m , -)	~~(m , -)~~ (ž , 4)	~~(m , -)~~ (ž , 4)

Primer

Generisati mašinski kod za sledeću instrukciju :

d := (a – b) + (a – c) + (a – c).

Međukod :

1. t := a – b
2. u := a – c
3. v := t + u
4. d := v + u

v = (ž , 4)
d = (ž , -)

t = (m , -)
v = (m , -)

u = (ž , 4)
u = (m , -)

Tabela simbola

a	b	c	d	t	u	v
(ž , -)	(ž , -)	(ž , -)	~~(ž , -)~~	~~(m , -)~~	~~(m , -)~~	~~(m , -)~~
			(m , -)	**(ž , 3)**	~~(ž , 4)~~	~~(ž , 4)~~
					(ž , 3)	**(m , -)**

Primer

Generisati mašinski kod za sledeću instrukciju :

d := (a – b) + (a – c) + (a – c).

Međukod :

1. t := a – b
2. u := a – c
3. v := t + u
4. d := v + u

u = (ž , 3)	**a = (ž , -)**	**c = (ž , -)**
v = (ž , 4)	t = (m , -)	u = (ž , 4)
d = (ž , -)	v = (m , -)	u = (m , -)

Tabela simbola

a	b	c	d	t	u	v
(ž , -)	(ž , -)	(ž , -)	~~(ž , -)~~	~~(m , -)~~	~~(m , -)~~	~~(m , -)~~
			(m , -)	(ž , 3)	~~(ž , 4)~~	~~(ž , 4)~~
					(ž , 3)	(m , -)

Primer

Generisati mašinski kod za sledeću instrukciju :

d := (a – b) + (a – c) + (a – c).

Međukod :

1. t := a – b
2. u := a – c
3. v := t + u
4. d := v + u

u = (ž , 3)
v = (ž , 4)
d = (ž , -)

a = (ž , -)
t = (m , -)
v = (m , -)

c = (ž , -)
u = (ž , 4)
u = (m , -)

Tabela simbola

a	b	c	d	t	u	v
~~(ž , -)~~	(ž , -)	~~(ž , -)~~	~~(ž , -)~~	~~(m , -)~~	~~(m , -)~~	~~(m , -)~~
(ž , 2)		**(ž , 2)**	(m , -)	(ž , 3)	~~(ž , 4)~~	~~(ž , 4)~~
					~~(ž , 3)~~	(m , -)
					(m , -)	

Primer

Generisati mašinski kod za sledeću instrukciju :

d := (a – b) + (a – c) + (a – c).

Međukod :

1. t := a – b	**t = (ž , 3)**	**a = (ž , 2)**	**b = (ž , -)**
2. u := a – c	u = (ž , 3)	a = (ž , -)	c = (ž , -)
3. v := t + u	v = (ž , 4)	t = (m , -)	u = (ž , 4)
4. d := v + u	d = (ž , -)	v = (m , -)	u = (m , -)

Tabela simbola

a	b	c	d	t	u	v
~~(ž , -)~~	(ž , -)	~~(ž , -)~~	~~(ž , -)~~	~~(m , -)~~	~~(m , -)~~	~~(m , -)~~
(ž , 2)		(ž , 2)	(m , -)	(ž , 3)	~~(ž , 4)~~	~~(ž , 4)~~
					~~(ž , 3)~~	(m , -)
					(m , -)	

Primer

Generisati mašinski kod za sledeću instrukciju :

d := (a – b) + (a – c) + (a – c).

Međukod :

1. t := a – b	t = (ž , 3)	a = (ž , 2)	b = (ž , -)
2. u := a – c	u = (ž , 3)	a = (ž , -)	c = (ž , -)
3. v := t + u	v = (ž , 4)	t = (m , -)	u = (ž , 4)
4. d := v + u	d = (ž , -)	v = (m , -)	u = (m , -)

Tabela simbola

a	b	c	d	t	u	v
~~(ž , -)~~	~~(ž , -)~~	~~(ž , -)~~	~~(ž , -)~~	~~(m , -)~~	~~(m , -)~~	~~(m , -)~~
~~(ž , 2)~~	**(ž , 1)**	(ž , 2)	(m , -)	~~(ž , 3)~~	~~(ž , 4)~~	~~(ž , 4)~~
(ž , 1)				**(m , -)**	~~(ž , 3)~~	(m , -)
					(m , -)	

Strukture podataka generatora koda

- Generator koda koristi desktriptor registra za svaki registar i desktriptor adresa za svaku promenljivu.

  Deskriptor registara sadrži spisak imena promenljivih čije vrednosti se nalaze u pojedinim registrima (na početku bloka prazan).

  Primer: AX : a, b, c

  Deskriptor adresa pokazuje u kojim se sve registrima i memorijskim lokacijama nalazi vrednost neke promenljive (na početku bloka sve korisničke prom. su u memoriji)

  Primer: a : AX, BX, a’

Algoritam generisanja koda za i) x := y op z

1. Poziva se potprogram **getreg** da bi se dobila lokacija **L** u koji će biti smešten rezultat **x**. Obično funkcija vraća registar, a ređe fizičku lokaciju operativne memorije
2. Ako **y** nije u **L** onda se generiše mašinska instrukcija:

   MOV L, y'

   gde je **y'** tekuća lokacija za **y** (najbolje ako je neki registar).
3. Generiše se mašinska instrukcija

   op L, z'

   gde je **z'** tekuća lokacija za **z** (najbolje ako je neki registar).
4. Dodati **L** u adresni deskriptor za **x**. Ako je **L** registar, dodati **x** u reg. deskriptor za **L**.
5. Ako **y** i **z** nisu živi u i-toj četvorci, udaljiti ih iz deskriptora registara u kojima se trenutno nalaze.
6. Posle obrade četvorki, na kraju bazičnog bloka, generisati instrukcije:

   MOV w', R

   za sve promenljive **w** koje su žive, a nalaze se isključivo u nekom od registara **R**

Potprogram getreg

Ulaz: *četvorka* ***x:=y op z ,*** *fleg da li je neophodan registar*

Izlaz: *registar* ***R*** *ili mem. lokacija za* ***y***

1. Ako je **y** jedina promenljiva u nekom registru **R** i **y** je mrtvo u posmatranoj četvorci, izbaciti **R** iz adr. deskriptora za **y** i vratiti **R**
2. Ako (1) ne uspe, vratiti prazan registar (ako ima takav).
3. Ako (2) ne uspe, i ako **x** ima sledeću upotrebu u tom bloku ili ako je **op** operator koji obavezno zahteva registar (npr. indeksiranje) pronaći neki zauzeti registar. Sa

   MOV w', R

   upisati u memoriju vrednosti svih promenljivih **w** iz **R** koje nisu u memoriji, izbaciti **R** iz adr. deskriptora tih promenljivih i vratiti **R**.
4. Ako tačka (3) ne uspe, vratiti memorijsku lokaciju za **x**.

Generisanje koda za tekući primer

četvorke	mašinski kod	Deskr. registara	Adresni deskriptori
		registri prazni	a, b, c i d u memoriji
t := a - b			
u := a - c			
v := t + u			
d := v + u			

- Pretpostavimo da imamo na raspolaganju dva registra, AX i BX.
- Mrtve promenljive su podvučene

Generisanje koda za tekući primer

četvorke	mašinski kod	Deskr. registara	Adresni deskriptori
		registri prazni	a, b, c i d u memoriji
t := a - b	**MOV** AX, a' **SUB** AX, b'		
u := a - c			
v := t + u			
d := v + u			

Generisanje koda za tekući primer

četvorke	mašinski kod	Deskr. registara	Adresni deskriptori
		registri prazni	a, b, c i d u memoriji
t := a - b	**MOV** AX, a' **SUB** AX, b'	AX sadrži t	t u AX, a, b, c u mem.
u := a - c			
v := t + u			
d := v + u			

Generisanje koda za tekući primer

četvorke	mašinski kod	Deskr. registara	Adresni deskriptori
		registri prazni	a, b, c i d u memoriji
t := a - b	**MOV** AX, a' **SUB** AX, b'	AX sadrži t	t u AX, a, b, c u mem.
u := a - c	**MOV** BX, a' **SUB** BX, c'		
v := t + u			
d := v + u			

Generisanje koda za tekući primer

četvorke	mašinski kod	Deskr. registara	Adresni deskriptori
		registri prazni	a, b, c i d u memoriji
t := a - b	**MOV** AX, a' **SUB** AX, b'	AX sadrži t	t u AX, a, b, c u mem.
u := a - c	**MOV** BX, a' **SUB** BX, c'	AX sadrži t BX sadrži u	t u AX, a, b, c u mem. u u BX
v := t + u			
d := v + u			

Generisanje koda za tekući primer

četvorke	mašinski kod	Deskr. registara	Adresni deskriptori
		registri prazni	a, b, c i d u memoriji
t := a - b	**MOV** AX, a' **SUB** AX, b'	AX sadrži t	t u AX, a, b, c u mem.
u := a - c	**MOV** BX, a' **SUB** BX, c'	AX sadrži t BX sadrži u	t u AX, a, b, c u mem. u u BX
v := t + u	**ADD** AX, BX		
d := v + u			

Generisanje koda za tekući primer

četvorke	mašinski kod	Deskr. registara	Adresni deskriptori
		registri prazni	a, b, c i d u memoriji
t := a - b	**MOV** AX, a' **SUB** AX, b'	AX sadrži t	t u AX, a, b, c u mem.
u := a - c	**MOV** BX, a' **SUB** BX, c'	AX sadrži t BX sadrži u	t u AX, a, b, c u mem. u u BX
v := t + u	**ADD** AX, BX	AX sadrži v BX sadrži u	u u BX, a, b, c u mem. v u AX
d := v + u			

Generisanje koda za tekući primer

četvorke	mašinski kod	Deskr. registara	Adresni deskriptori
		registri prazni	a, b, c i d u memoriji
t := a - b	**MOV** AX, a' **SUB** AX, b'	AX sadrži t	t u AX, a, b, c u mem.
u := a - c	**MOV** BX, a' **SUB** BX, c'	AX sadrži t BX sadrži u	t u AX, a, b, c u mem. u u BX
v := t + u	**ADD** AX, BX	AX sadrži v BX sadrži u	u u BX, a, b, c u mem. v u AX
d := v + u	**ADD** AX, BX		

Generisanje koda za tekući primer

četvorke	mašinski kod	Deskr. registara	Adresni deskriptori
		registri prazni	a, b, c i d u memoriji
t := a - b	**MOV** AX, a' **SUB** AX, b'	AX sadrži t	t u AX, a, b, c u mem.
u := a - c	**MOV** BX, a' **SUB** BX, c'	AX sadrži t BX sadrži u	t u AX, a, b, c u mem. u u BX
v := t + u	**ADD** AX, BX	AX sadrži v BX sadrži u	u u BX, a, b, c u mem. v u AX
d := v + u	**ADD** AX, BX	AX sadrži d	d u AX, a, b, c u mem.

Generisanje koda za tekući primer

četvorke	mašinski kod	Deskr. registara	Adresni deskriptori
		registri prazni	a, b, c i d u memoriji
t := a - b	**MOV** AX, a' **SUB** AX, b'	AX sadrži t	t u AX, a, b, c u mem.
u := a - c	**MOV** BX, a' **SUB** BX, c'	AX sadrži t BX sadrži u	t u AX, a, b, c u mem. u u BX
v := t + u	**ADD** AX, BX	AX sadrži v BX sadrži u	u u BX, a, b, c u mem. v u AX
d := v + u	**ADD** AX, BX	AX sadrži d	d u AX, a, b, c u mem.
	MOV d', AX	registri prazni za novi bazični blok	a, b, c u mem.

Pravila generisanja koda za pristup elementima niza:

Naredba	i je u registru R		i je u memoriji M		i je na steku, pomeraj di	
	Kod	Cena	Kod	Cena	Kod	Cena
a:=b[i]	MOV R', [b+R]	2	MOV R, M MOV R, [R+b]	4	MOV R, [BP+di] MOV R, [R+b]	4
a[i]:=b	MOV [a+R], b	3	MOV R, M MOV [R+a], b	5	MOV R, [BP+di] MOV [R+a], b	5

Pravila generisanja koda pri upotrebi pokazivača:

Naredba	p je u registru R		p je u memoriji M		p je na steku,pomeraj dp	
	Kod	Cena	Kod	Cena	Kod	Cena
a:=*p	MOV a, [R]	2	MOV R, M MOV R, [R]	3	MOV R, [BP+dp] MOV R, [R]	3
*p:=a	MOV [R], a	2	MOV R, M MOV [R], a	4	MOV R, [BP+dp] MOV [R], a	4