A Simple Code Generator

A Simple Code Generator

- Assumption:
  - Input : Three address code
- Code generator generates code for single basic block
- For each operator, one machine instruction that takes necessary operands in registers and performs the operation and leaves the result in a register
  - LD reg, mem
  - ST mem, reg
  - OP reg, reg, reg

A Simple Code Generator

- Assumption: An algorithm makes use of template code for each three address instruction
- Example
  - z=x+y

```
LD R1, x
LD R2, y
ADD R3, R2, R1
ST z, R3
```

A Simple Code Generator

- c=a+b
- e=c+d
  - LD R1, a
  - LD R2, b
  - ADD R3, R1, R2
  - ST c, R3
  - LD R4, c
  - LD R5, d
  - ADD R6, R4, R5
  - ST e, R6
- A code generator alg. Should check, before issuing a LD, whether the value to be loaded is already in a register.

Register and Address Descriptors

- Need of data structure : Information about what program variables currently have their value in a register and which register.
  - **Register Descriptor:** Keeps track of the variable names whose current value is in that register
  - **Address Descriptor:** Keeps track of the location where the current value of that variable is found.

Managing Register and Address Descriptors

- LD R,x
  - Change the register descriptor for Register R so it holds only x
  - Change the address descriptor for x by adding register R as an additional location
- ST x, R
  - Change the address descriptor for x to include its own memory location
- ADD Rx, Ry, Rz (x=y+z)
  - Change register descriptor for Rx to hold x
  - Change the address descriptor of x so that its only location is Rx
  - Remove Rx from the address descriptor of any variable other than x
- Copy Statement (x=y)
  - Add x to the register descriptor for Ry
  - Change the address descriptor of x so that its only location is Ry

Example

t=a-b

u=a-c

v=t+u

a=d

d=v+u

Example

t=a-b

LD R1, a

LD R2, b

SUB R2, R1, R2

Initial

R1	R2	R3

a	b	c	d	t	u	v
a	b	c	d			

R1	R2	R3
a	t	

a	b	c	d	t	u	v
a, R1	b	c	d	R2		

After

Example

u=a-c

LD R3, c

SUB R1, R1, R3

Initial

R1	R2	R3
a	t	

a	b	c	d	t	u	v
a, R1	b	c	d	R2		

R1	R2	R3
u	t	c

a	b	c	d	t	u	v
a	b	c, R3	d	R2	R1	

After

Example

v=t+u

ADD R3, R2, R1

Initial

R1	R2	R3
u	t	c

a	b	c	d	t	u	v
a	b	c, R3	d	R2	R1	

R1	R2	R3
u	t	v

a	b	c	d	t	u	v
a	b	c	d	R2	R1	R3

After

Example

a=d

LD R2, d

Initial

R1	R2	R3
u	t	v

a	b	c	d	t	u	v
a	b	c	d	R2	R1	R3

R1	R2	R3
u	a,d	v

a	b	c	d	t	u	v
R2	b	c	D, R2		R1	R3

After

Example

d=v+u

ADD R1, R3, R1

Initial

R1	R2	R3
u	a,d	v

a	b	c	d	t	u	v
R2	b	c	d, R2		R1	R3

R1	R2	R3
d	a	v

a	b	c	d	t	u	v
R2	b	c	R1			R3

After

Example

exit

ST a, R2

ST d, R1

Initial

R1	R2	R3
d	a	v

a	b	c	d	t	u	v
R2	b	c	R1			R3

R1	R2	R3
d	a	v

a	b	c	d	t	u	v
a, R2	b	c	d,R1			R3

After

Reference

- Aho A.V., Lam M.S., Sethi R., and Ullman J.D. Compilers: Principles, Techniques, and Tools (ALSU). Pearson Education, 2007.