COMPILADORES INSTRUÇÔES

QUALQUER DUVIDA QUE APARECER MANDE UM EMAIL PARA: nunesfrancisco927@gmail.com

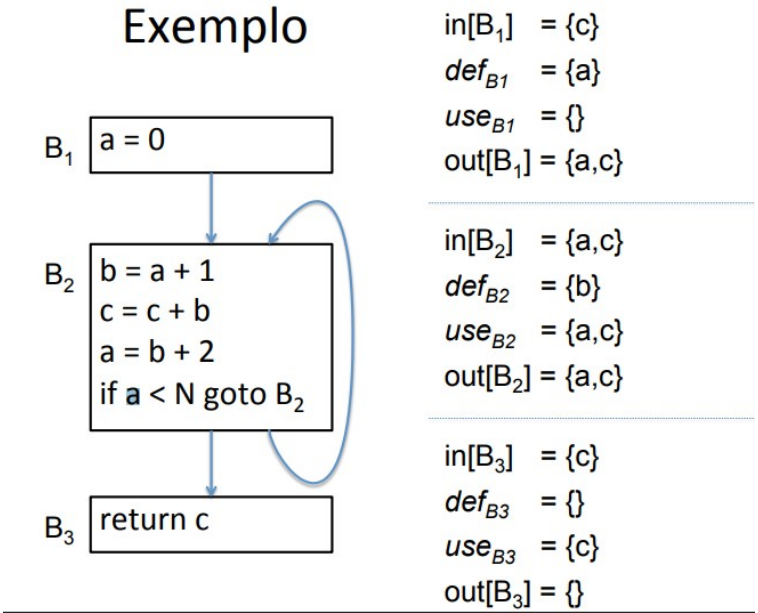
Este arquivo tem como objetivo mostrar como rodar o trabalho 2 de compiladores, mostrar como fazer toda a entrada e como ver as saidas. A opção do trabalho escolhida foi o 3, implementado a Analise de Longevidade, de Reaching Definitions e Avaliable Expressions.

ENTRADA:

A entrada do programa é feita da seguinte forma, quando rodar o programa, vai aparecer na tela oque está pedindo para ser escrito, ele começa pedindo a entrada de N, N é o numero total de blocos, depois é pedido o valor de M, M é o número de linhas do bloco, a cada bloco vai ser pedido uma entrada de M, para poder referenciar o número especifico de cada linha do bloco.
ATENÇÂO, o número do id de cada bloco já é decidido automaticamente, então ele segue a ordem de que o primeiro bloco que adicionar vai ser o B1, o segundo o B2, e assim por diante.
Depois de ter adicionado N e M, vai ser pedido o código, linha por linha, o numero de códigos adicionados é igual a M, como mencionado antes. Como exemplo, a = b + c seria um exemplo de código adicionado em apenas uma linha, mais a frente tem um exemplo mais claro de uma entrada.
Depois de adicionar o código, aparece pedindo para dizer a quantos blocos o bloco que acabou de ser montado vai ser ligado, e depois é necessario referenciar o número desses blocos, os números dos blocos iniciam e 1 e não em 0.
E todo esse processo é necessario ser feito para todos os N blocos.
Não existe mudança na entrada entre os 3 algoritmos, mas é necessario tomar cuidado, pois cada algoritmo foi montado pensando que a entrada para aquele algoritmo está satisfeita, mais a frente também tem exemplos sobre isso.
O código utiliza duas struct, uma para fazer os blocos, e outra para guardar os resultados de in, def, use e out dos algoritmos, nos que utilizam o kill, decidi deixar o use, para representar ele pelo código e não precisar criar mais uma struct.

ANALISE DE LONGEVIDADE:

Vamos utilizar este exemplo para a analise de longevidade.
Como entrada teriamos:
N: 3
M: 1
codigo: a = 0
pra quantos aponta: 1
quais blocos ele aponta: 2
M: 4
codigo: b = a + 1
codigo: c = c + b
codigo: a = b + 2
codigo: if a < N goto B2
pra quantos aponta: 2
quais blocos ele aponta: 2
quais blocos ele aponta: 3
M: 1
codigo: return c
pra quantos aponta: 0
essa seria a entrada nesse caso.



COMPILADORES INSTRUÇÔES

NOTA: Esse algoritimo foi totalmente implementado para caso uma das variaveis não tenha sido definida entre os blocos, ela diz que essa variavel veio de um bloco a cima do B1, como no exemplo, a resposta do código vai ter exatamente A MESMA SAIDA, com o In do B1 tendo c.

Caso não queria ter todo o trabalho de digitar linha a linha, pode copiar e colar isso no terminal e apertar enter(uma ou dias vezes), segue todas as entradas direitinho para o exemplo de cima. Se infelizmente isso não funcionar, vai ter que passar cada um linha a linha.

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
3
1
a = 0
1
2
4
b = a + 1
c = c + b
a = b + 2
if a < N goto B2
2
2
3
1
return c
0
```

e teremos nossa saida:

```
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Analise de Longevidade!
B1: in = {c,}
B1: def = {a,}
B1: use = {}
B1: out = {a,c}
B2: in = {a,c,}
B2: def = {b,}
B2: use = {a,c,}
B2: out = {a,c}
B3: in = {c,}
B3: def = {}
B3: use = {c,}
B3: out = {}
```

Por favor, ignore as virgulas a mais no codigo, não tive tempo de criar um codigo especifico para tirar cada uma delas.

Sobre o código:

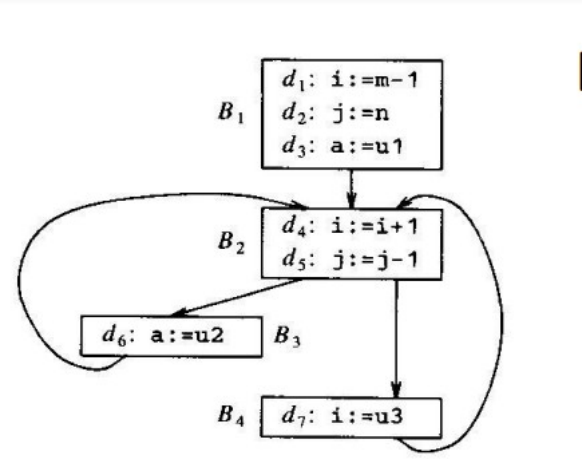
O código da analise de longevidade começa na linha 22 e termina na linha 152, ele é dividido em 5 códigos, o primeira é o do algoritmo de Use(linhas 22 até 57), depois tem o código do que ele Define(linhas 59 até 71), depois temos o código do Out(linhas 73 até 94), depois o código do In(linhas 96 até 130) e por ultimo, temos o código que chama todas essas funções, para assim calcular a longevidade(linhas 132 até 152). Essa função é chamada na linha 366, e nas linhas 370

COMPILADORES INSTRUÇÔES

até 376 é onde está printando o resultado, caso queira apenas ver o resultado de longevidade, pode comentar os outros dois for que tem logo em baixo.

REACHING DEFINITIONS:

Vamos pegar esse exemplo para calcular o Reaching Definitions, teremos a entrada:
N: 4
M: 3
codigo: i = m – 1
codigo: j = n
codigo: a = u1
pra quantos aponta: 1
quais blocos ele aponta: 2
M: 2
codigo: i = i + 1
codigo: j = j - 1
pra quantos aponta: 2
quais blocos ele aponta: 3
quais blocos ele aponta: 4
M: 1
codigo: a = u2
pra quantos aponta: 1
quais blocos ele aponta: 2
codigo: i = u3
pra quantos aponta: 1
quais blocos ele aponta: 2



B	IN[B]	OUT[B]
1		d1, d2, d3
2	d1, d2, d3, d6	d3, d4, d5, d6
3	d3, d4, d5, d6	d4, d5, d6
4	d3, d4, d5, d6	d3, d5, d6, d7

Nesse código, tudo precisa ser bem definido em cada bloco, ou seja, se tentarmos querer que ele pegue algo que foi definido em um bloco a fora, já não vai funcionar, pois o algoritmo construido não foi feito tendo isso em vista, ele apenas considera tudo que foi declarado nos blocos criados, ou seja, se tentarmos colocar o exemplo de longevidade, ele nunca vai saber que um valor c foi criado.

Atenção, abaixo tem o exemplo para copiar e colar no terminal, mas infelizmente por algum motivo ele estava bugando quando colava no terminal e copiava apenas os tres primeiros valores, caso isso aconteça com o professor, infelizmente será necessario colocar a entrada linha a linha.
Nas saidas não estão em ordem, mas todas as saidas estão saindo corretas.

COMPILADORES INSTRUÇÔES

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
4
3
i = m – 1
j = n
a = u1
1
2
2
i = i + 1
j = j - 1
2
3
4
1
a = u2
1
2
i = u3
1
2
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Reaching Definitions!
B1: in = {}
B1: gen = {d1,d2,d3}
B1: kill = {d4,d7,d5,d6}
B1: out = {d1,d2,d3}
B2: in = {d1,d2,d3,d6,d7}
B2: gen = {d4,d5}
B2: kill = {d1,d7,d2}
B2: out = {d4,d5,d3,d6}
B3: in = {d4,d5,d3,d6}
B3: gen = {d6}
B3: kill = {d3}
B3: out = {d6,d4,d5}
B4: in = {d4,d5,d3,d6}
B4: gen = {d7}
B4: kill = {d1,d4}
B4: out = {d7,d5,d3,d6}
```

Sobre o código:
O código de reaching definition assim como o de longevidade, está dividido em, gera(linhas 156 até 166), kill(linhas 168 até 185), e o in e o out(nas linhas 187 até 203), uma função auxiliar para tirar as virgulas(linhas 205 até 219), esse foi o único código que tive tempo de fazer isso. E para acabar, a função que chama todas elas(linhas 221 até 239). E ela é chamada na linha 367, com print nas linhas 377 até 383.

COMPILADORES INSTRUÇÔES

AVALIABLE EXPRESSIONS:

Peguemos esse exemplo para Expressoes avaliaveis, vamos considerar isso apenas como um bloco:

N: 1
M: 4
codigo: a = b + c
codigo: b = a - d
codigo: c = b + c
codigo: d = a - d
pra quantos aponta: 0

>>>>>>>>>>>>>>>>>>>>>>>>>>>>
1
4
a = b + c
b = a - d
c = b + c
d = a - d
0
<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Avaliable Expressions!
B1: in = {}
B1: gen = {b+c,a-d,b+c,a-d,}
B1: kill = {abcd}
B1: out = {,,}

Available Expressions

Instruções	Expressões disponíveis
a = b + c	
b = a - d	
c = b + c	
d = a - d	

Nessa também não tive tempo de arrumar as saidas, então na parte de kill, considere que cada letra está separada por uma virgula, se seria referente a o, ele gera a expressão b + c e mata todas as ocorrencias de a, e isso está em ordem, e tambem é exatamente sobre isso esse kill, tudo que está antes do igual, ele pega como oque mata.

Sobre o codigo, assim como o outro, ele tem 4 codigos, um que define(linhas 243 até 259), um que mata(261 até 269), um de in e out(linhas 271 até 316) e um que chama todos eles(linha 319 até 327).A função é chamada na linha 368, que printa o resultado nas linhas 384 até 390.