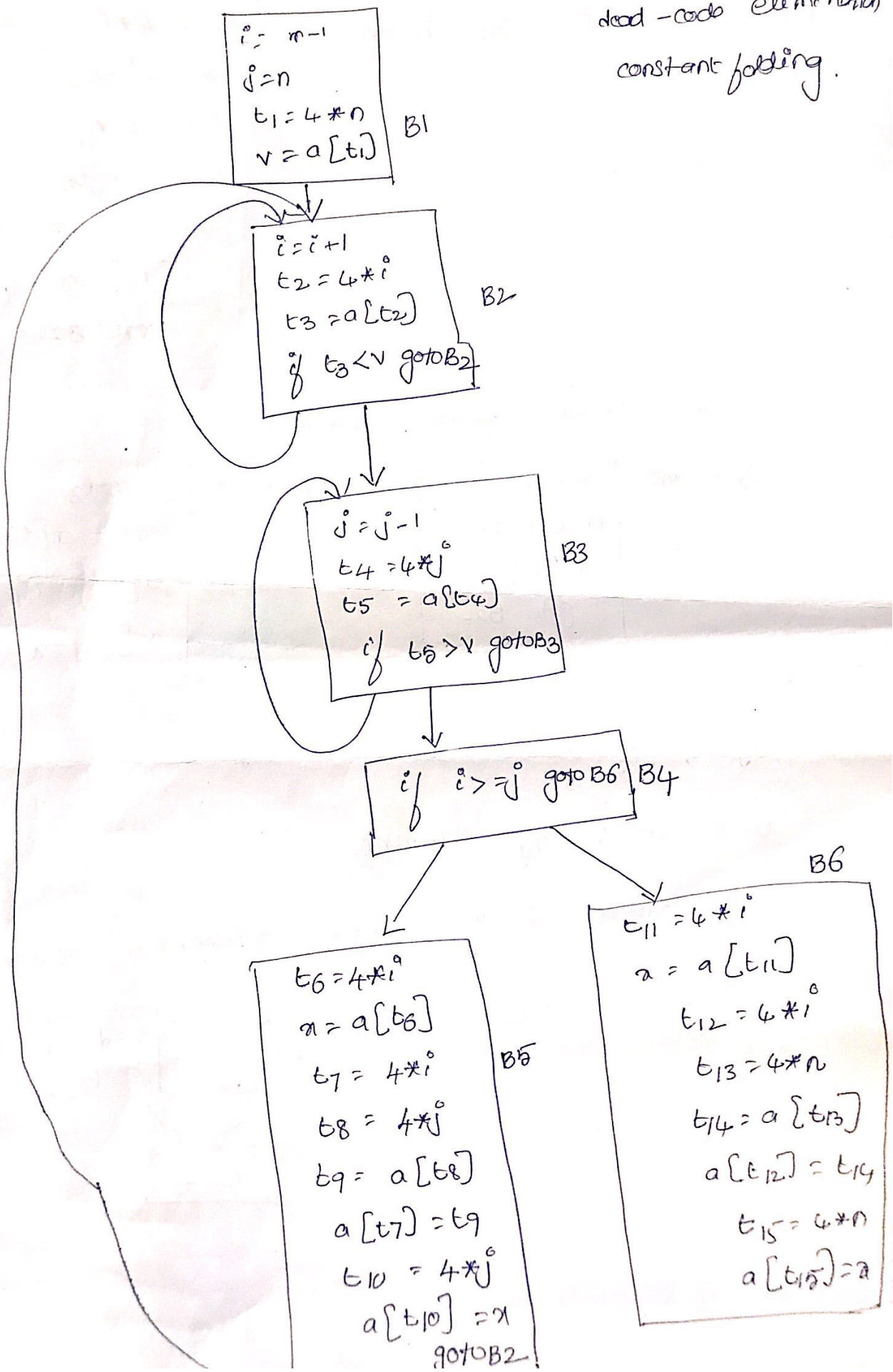
Semantic preserving transformation → common subexpr elimination (1)
copy propagation
dead-code elimination
constant folding.



Local common subexpression (L.C.S.E)

After L.C.S.E B5 becomes

```
t6 = 4*i
x = a[t6]
t8 = 4*j
t9 = a[t8]
a[t6] = t9
a[t8] = x
goto B2
```

Global common common subexpression

B5 becomes

```
x = t3
a[t2] = t5
a[t4] = x
goto B2
```

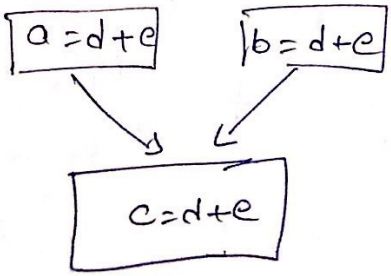
B6 becomes

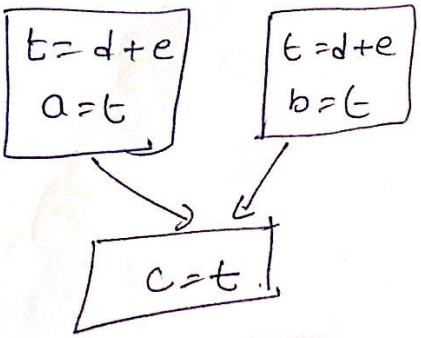
```
x = t3
t14 = a[t4]
a[t2] = t14
a[t1] = x
```

copy propagation

U = V called copy statements.

copies introduced during common subexpression elimination.





Basic block B5 after copy propagation

```
x = t3
a[t2] = t5
a[t4] = t3
goto B2
```

Dead - code elimination

if (debug) print if debug = false it won't enter into loop.

one adv of copy propagation is often turns the copy stmt into dead code.

B5 still reduces to

```
a[t2] = t5
a[t4] = t3
goto B2
```

code motion

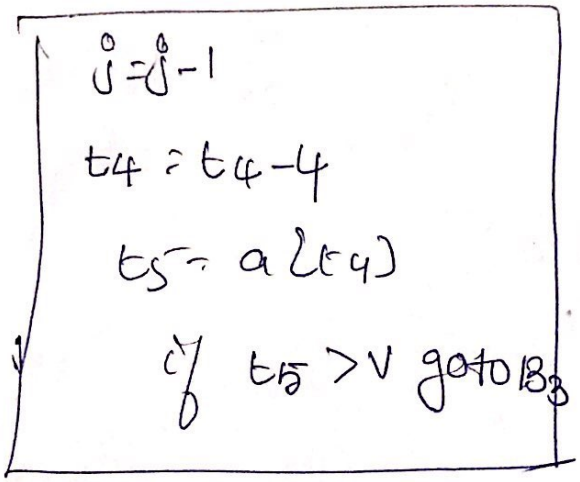
Loop invariant computation - same result independent of the no of times a loop is executed.

while (i <= limit - 2)

```
t = limit - 2
while (i <= t)
```

B3

Induction variables and Reduction in strength is



if (cond)

then A =

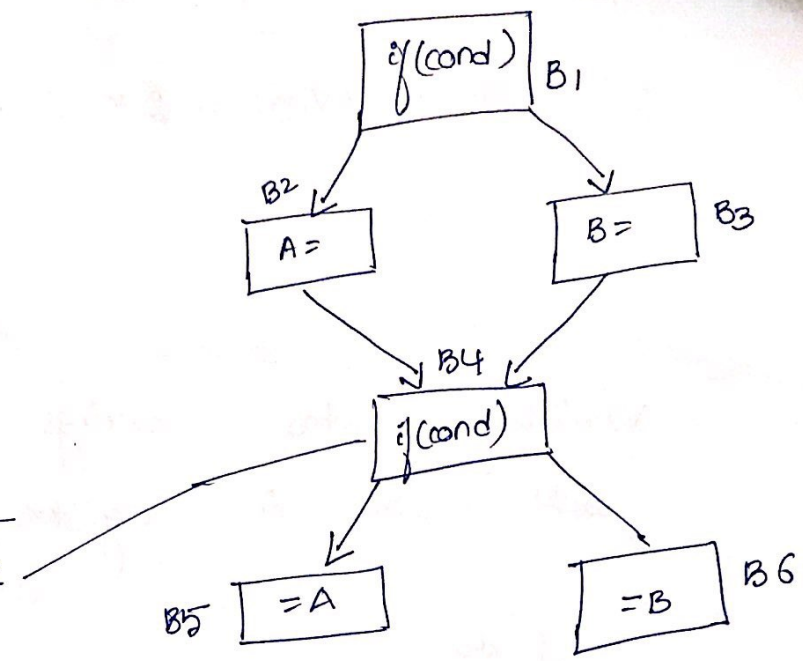
else B =

X : if (cond)

then = A

else = B

A and B both live



Global Register Allocation via usage counts

1. for every usage of a variable V in a BB until it is first defined do:

- savings (v) = savings (v) + 1
- after v is defined, it stays in the register any way and all further references are to that register.

2. for every variable V computed in a BB, if it is live on exit from the BB.

count a savings of 2, since it is not necessary to store it at the end of the BB

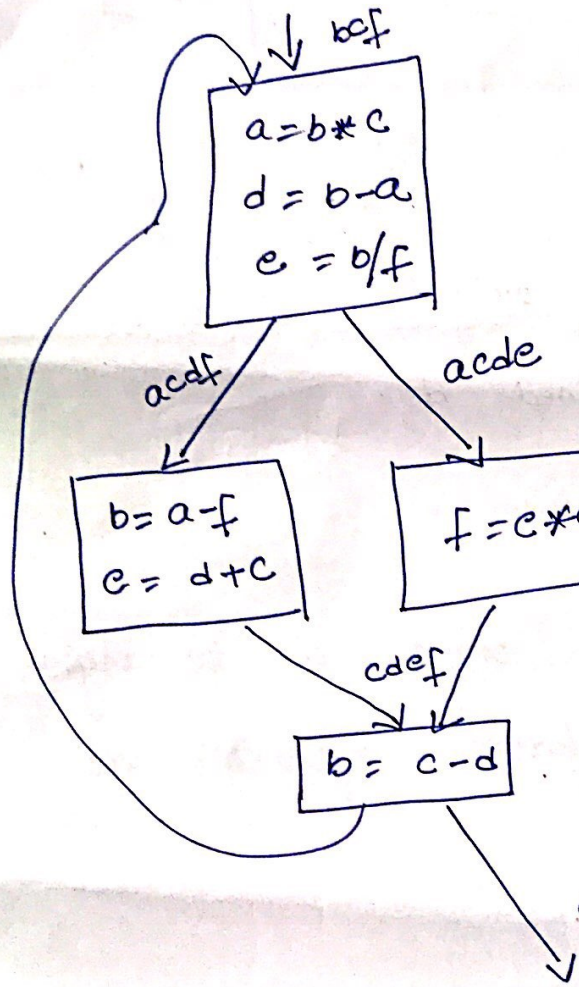
Total savings per variable V are

Σ (savings (V, B) + 2 * live and computed (V, B))

↓

it may be 0 or 1

- Variable whose savings are the highest will reside in registers



saving for the variable

a: (0+2) + (1+0) + (1+0) + (0+0) = 4

b: (3+0) + (0+0) + (0+0) + (0+2) = 5

c: (1+0) + (1+0) + (0+0) + (1+0) = 3

d: (0+2) + (1+0) + (0+0) + (1+0) = 4

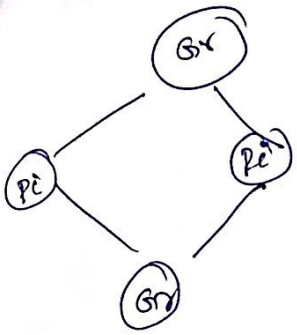
e: (0+2) + (0+2) + (1+0) + (0+0) = 5

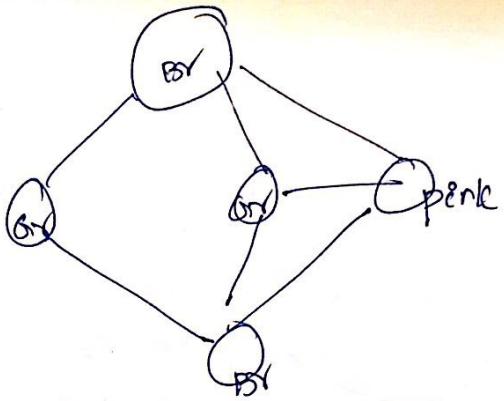
f: (1+0) + (1+0) + (0+2) + (0+0) = 4

a, b, e will be allocated to the variable

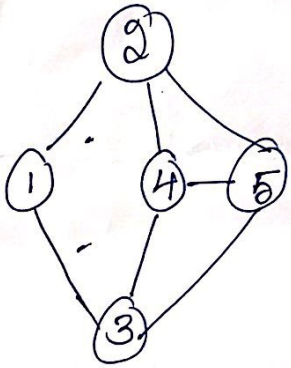
Graph coloring

three colourable

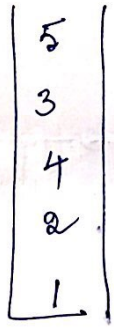




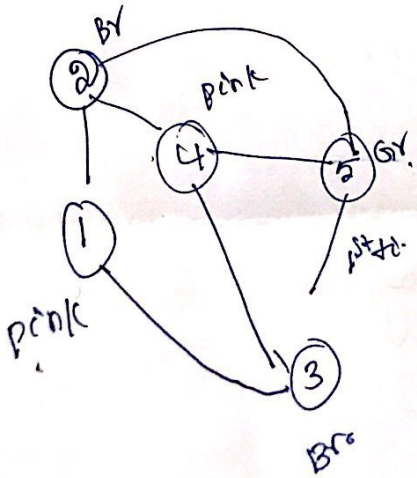
chose an arbitrary node of degree less than K and put it on the stack.



Registers

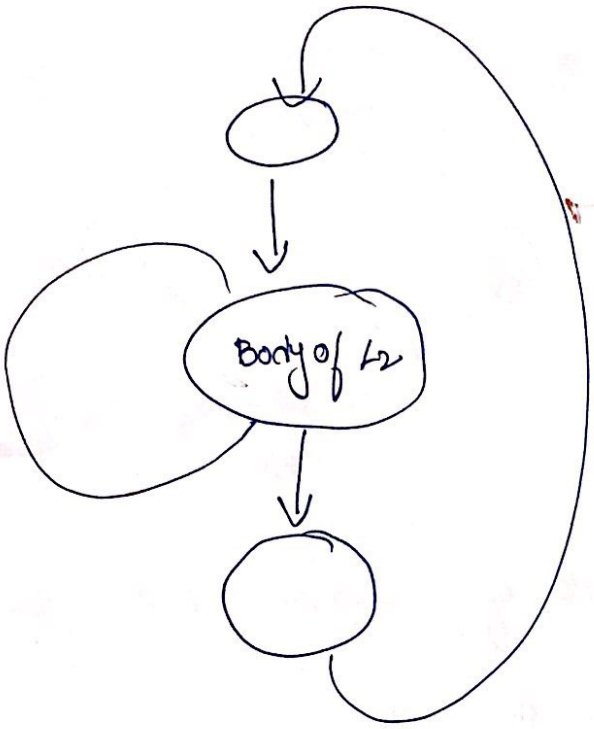


Stack



3 registers

For nested loop



case 1 : variables x, y, z assigned registers in L2, but not in L1

Load x, y, z on entry to L2

Store x, y, z on exit from L2

case 2 : variables a, b, c assigned registers in L1, but not in L2

Store a, b, c on entry to L2

Load a, b, c on exit to L2

case 3 : variables P, Q assigned registers in both L1 and L2.

Machine - Independent optimizations.

Local code optimization — code improvement with in basic block.

Global code optimization — improvements to a program which happens across basic blocks.

— based data flow Analysis.

principle source of optimization.

9444050188

semantics - preserving transformations.

Global common subexpression elimination

copy propagation

dead code elimination

code motion

Induction variables and Reduction in strength.

Foundations of data flow - Analysis.

DFA framework $(D, V, \wedge, F)$ consists of

1. A direction of the data flow D which is either FORWARDS (or) BACKWARDS.
2. A semilattice which includes a domain values of V and a meet operator $\wedge$.
3. A family F of transfer functions from V to V.

   this family must include functions suitable for the boundary conditions, which are constant transfer functions for the special nodes ENTRY and EXIT in any flow graph.

Semilattice

A semilattice is a set V and a binary meet operator $\wedge$ such that for all x, y and z in V.

1. $x \wedge x = x$ (meet is idempotent)
2. $x \wedge y = y \wedge x$ (" " commutative)
3. $x \wedge (y \wedge z) = (x \wedge y) \wedge z$ (" " associative)

A semilattice has a top element, denoted $\top$, such that for all x in V

$$\top \wedge x = x$$

optionally, a semilattice may have a bottom element denoted $\bot$ such that for all x in V, $\bot \wedge x = \bot$

partial orders

meet operator of a semilattic define a partial order on the values of the domain

a relation $\leq$ is a partial order on a set V if for all x, y and z in V:

1. $x \leq x$ (partial order is reflexive)
2. if $x \leq y$ and $y \leq x$ then $x = y$ (the partial order is antisymmetric)
3. if $x \leq y$ and $y \leq z$ then $x \leq z$ (partial order is transitive.

the pair $(V, \leq)$ is called a poset.

Monotone frame works

A frame work is monotone if when we apply any transfer function f in F to two members of V, the first being no greater than the second, then the first result is no greater than the second result.

A data-flow frame work is $(D, F, V, \wedge)$ is monotone if

for all x and y in V and f in F,

$x \leq y$ implies $f(x) \leq f(y)$

equivalently,

for all x and y in V and f in F,

$f(x \wedge y) \leq f(x) \wedge f(y)$.

Distributive frame work

$f(x \wedge y) = f(x) \wedge f(y)$.