What's in a Name?

OR

The Value of Renaming for Parallelism Detection and Storage Allocation

By

Himanshu (05305008)

Kamalakar (05305012)

Shrinadha(05305020)

Under guidance of

Prof. Supratim Biswas

Introduction

- Renaming : Introduction of an assignment to a new temporary variable in terms of given variable.
- Used to increase potential parallelism.
- Approaches studied yet.
- Removal of storage related dependencies.
- Anti-dependencies and Output-dependencies.

Eliminating Storage Related Dependencies

- Renaming uses Single Assignment Rule.
- Single Assignment Rule : Every variable defined exactly once.
- How to Rename?
- In a basic block, rename all definitions and uses of a variable except the final one.
- Guaranteed removal of all storage related dependencies.

Renaming with Basic Block

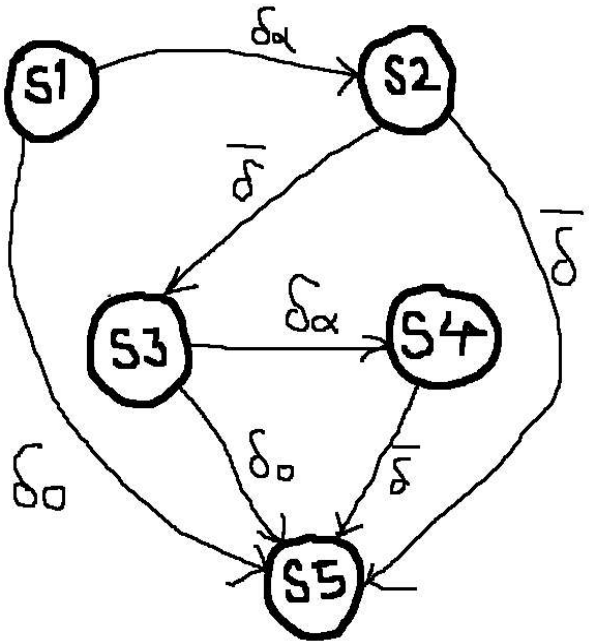
```
X  =  ...
   =  X
X  =  ...
   =  X
X  =  ...
```

Original Program

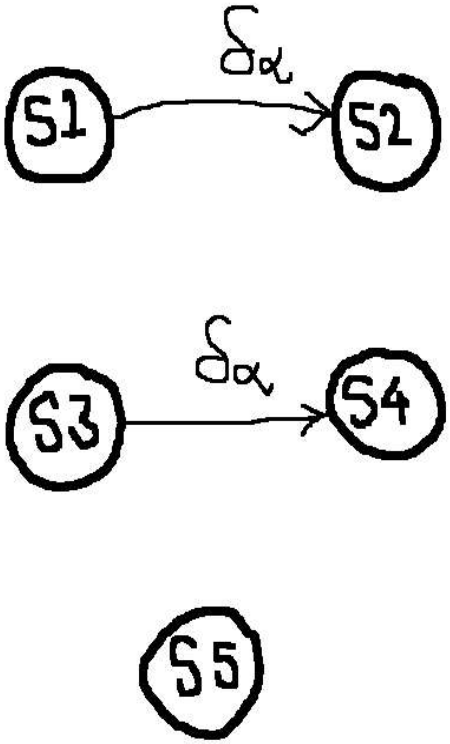
```
X1 =  ...
   =  X1
X2 =  ...
   =  X2
X  =  ...
```

Renamed Program

Data flow diagrams



Original Dependency



After Renaming

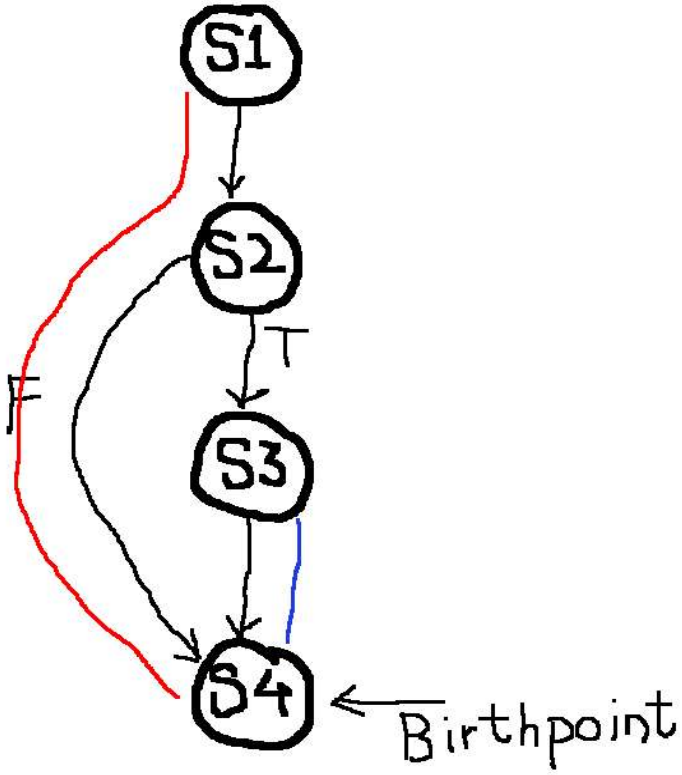
Eliminating Storage Related Dependencies (Contd.)

- Output-dependencies : Every variable defined only once.
- Anti-dependencies : Proof by contradiction.
  - Assume anti-dependency on variable X.
  - This implies that a use of X is before a definition.
  - However, the use above must be preceeded by some defination.
  - Contradiction : Two definitions!

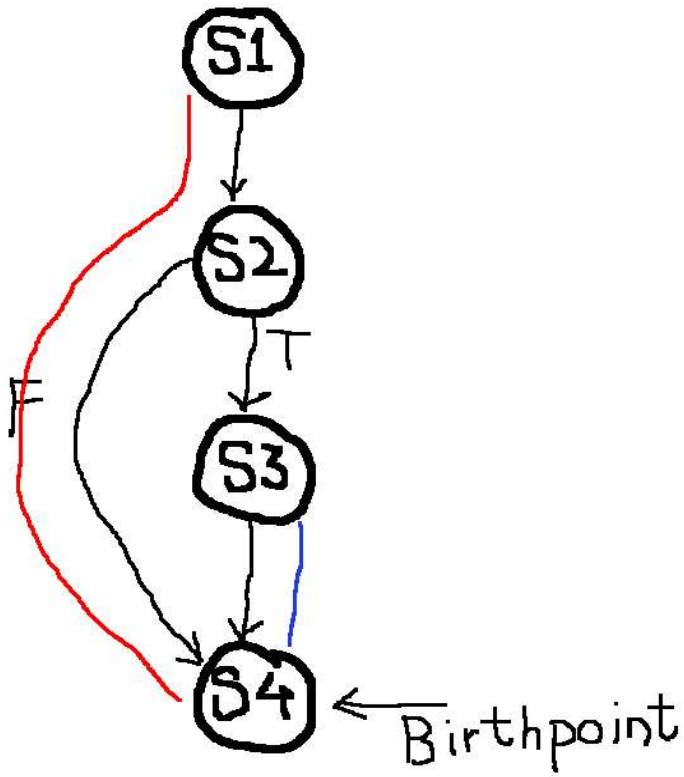
Renaming with If-Else construct

Join Birthpoint : Exists for a variable in the CFG where multiple definitions meet for the first time.

```
S1:     X = C + D
S2:     if ( P )
S3:     then X = A + B
S4:     Q = X
```



Renaming with If-Else construct (Contd.)




```
X1  =  C + D

if ( P )
then X2  =  A + B
        X   =  X2

else
        X   =  X1

Birthpoint (X)

Q  =  X
```

Renaming with Loops

For the purpose of renaming, loops can be thought of being divided into two types:

1. Bound not known at compile time.
2. Bound known at compile time.

Problem : Each definition and subsequent use of a variable within a loop requires a new name for each iteration.

Bound not known at compile time

Solution : Dynamic memory allocation.

```
DO while p
    X  =  X + A [ i ]
ENDDO
```

Original

```
DO while p
    NEW X  =  X + A [ i ]
ENDDO
```

Renamed

Bound known at compile time

Scalar Expansion : Scalar X is changed into an array sufficiently large to hold all the values assigned to X in the original program

```
DO i = 1 to 100
      X = 0                          S1
      if  ( p )  then X = 1          S2
                                     S3

      Z [ i ]  = X                   S4

      C [ i ]  = A [i + 1]           S5
      A [ i ]  = B [ i ]             S6

ENDDO

                                     S7
```

```
DO i = 1 to 100
      X0 [ i ] = 0
      if  ( p )     then X1  [ i ]  = 1
                    else  X1 [ i ]  = X0 [ i ]

Z [ i ]  =  X1 [ i ]

      C [ i ]  =  A1 [i + 1]
      A2 [ i ]  = B [ i ]

ENDDO

A [ * ] = A2 [ * ]
```

Storage Allocation with Renaming

- Main problem with Renaming : storage allocation for newly created variables.

- MAXLIVE : Max number of simultaneously live variables.
- MINCOLORS : Min number of storage locations needed.
- MAXLIVE = Lower_Bound (MINCOLORS)
- Using Renaming : MAXLIVE = MINCOLORS

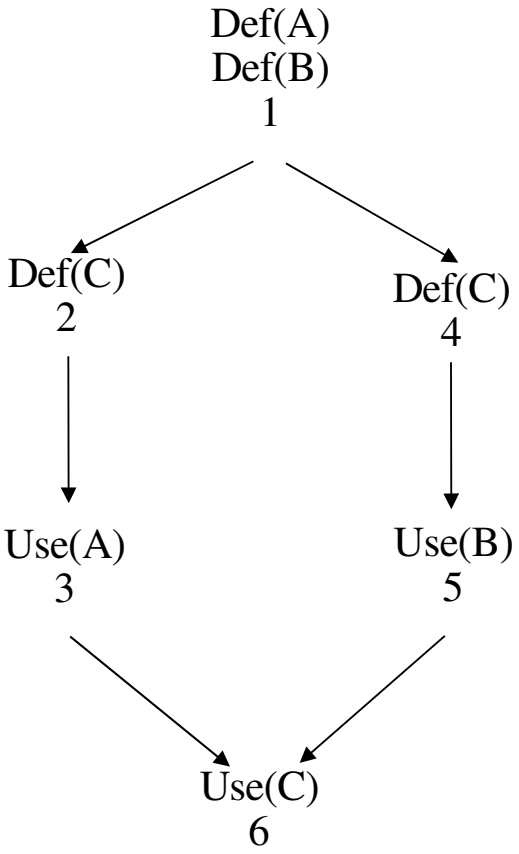
Storage Allocation with Renaming (contd.)

- Live Range of a variable V : set of nodes in the CFG where V is live.

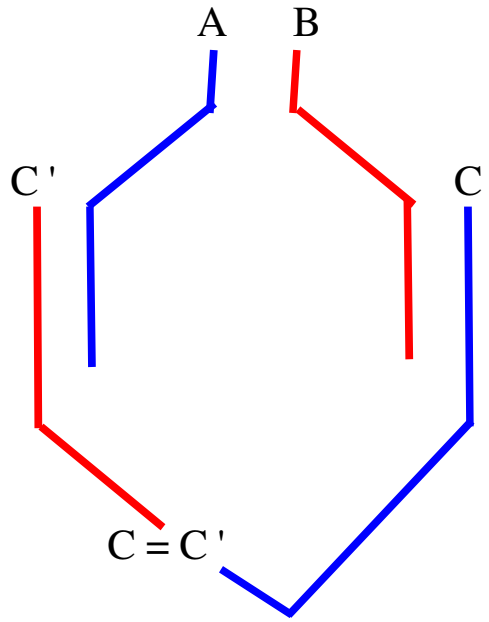
- Conflict : variables V and W conflicts if at the definition point of V, W is live.
- Coloring conflict : occurs between variables V and W if they conflict and both have been assigned same color.

Storage Allocation Algorithm

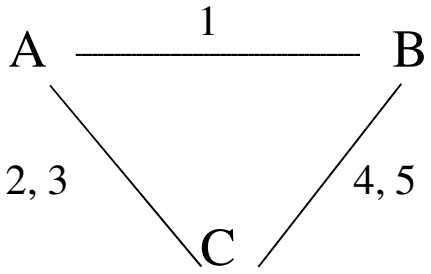
- Order all the definitions in the CFG in topological order, ignoring back edges.
- Start coloring the live ranges in order of their earliest definition point, using atmost MAXLIVE colors.
- Continue this process until a coloring conflict occurs.
- Whenever a coloring conflict occurs between V and W at the definition of V, rename the segment of the live range of V from the point of conflict to the next birthpoint of V by V'.



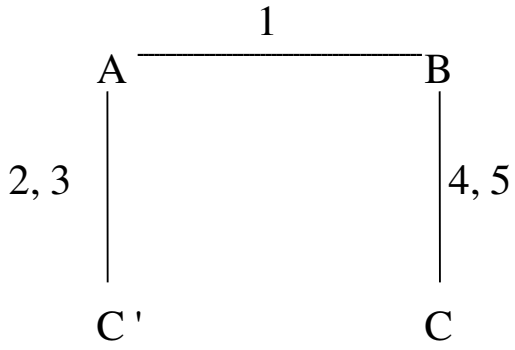
Before Renaming



Colored Live Ranges



Conflict Graph



Conflict Graph after Renaming

Conclusion

- Renaming guarantees elimination of all storage related dependencies.
- Increase in the degree of parallelism.
- Storage Allocation algorithm described needs atmost MAXLIVE extra storage locations.
- No previous Renaming approach has placed a limit on the amount of extra storage required.

Thank you