# Gimple Code

;; Function main (main, funcdef\_no=0, decl\_uid=1794, cgraph\_uid=0, symbol\_order=0)

;; 2 loops found

;;

;; Loop 0

;; header 0, latch 1

;; depth 0, outer -1

;; nodes: 0 1 2 3 4 5 6

;;

;; Loop 1

;; header 4, latch 3

;; depth 1, outer 0

;; nodes: 4 3

;; 2 succs { 4 }

;; 3 succs { 4 }

;; 4 succs { 3 5 }

;; 5 succs { 6 }

;; 6 succs { 1 }

main ()

{

int \* p;

int i;

int b[3];

int a[3];

int D.1804;

<bb 2> [0.00%]:

b[0] = 1;

b[1] = 2;

b[2] = 3;

i = 0;

goto <bb 4>; [0.00%]

<bb 3> [0.00%]:

\_1 = b[i];

a[i] = \_1;

i = i + 1;

<bb 4> [0.00%]:

if (i <= 2)

goto <bb 3>; [0.00%]

else

goto <bb 5>; [0.00%]

<bb 5> [0.00%]:

p = &a;

\_2 = p + 8;

\*\_2 = 5;

a = {CLOBBER};

b = {CLOBBER};

D.1804 = 0;

<L3> [0.00%]:

return D.1804;

}

# Code

int main()

{

int a[3];

int b[] = {1, 2, 3};

int i, \*p;

for (i=0; i<3; i++) {

a[i] = b[i];

}

p = a;

\*(p + 2) = 5;

}

# Answers

# 1]

The array is split into two parts one is initialization and the other deceleration in the body of the code. In array one element is declared in one statement at each time because in gimple we can perform one operation in one statement.

# 2]

Gimple can perform only one operation per statement hence it cannot read and write in the same statement. Therefore, the statement is split into two statement one to read and the another to write using temporary variable.

# 3]

CLOBBER statements are used for address-escaped variables. It is an annotation to signify that the scope of this address-escaped variable has ended.