School of Mathematical and Computational Sciences

Indian Association for the Cultivation of Science

Compiler Construction: COM 5202

Tutorial XIII (23 April, 2025)

M. Sc Semester IV: 2024-2025 *Instructor:* Goutam Biswas

Exercise 1. *Consider the following C function:*

```
void initMat(int a[][10], int n){
     int i, j;
     for(i=0; i<n; ++i)
         for(j=0; j<n; ++j) a[i][j]=1;
}
```

Corresponding 3-address code is as follows:

```
    i=0
    goto L4
L5:
    j=0
    goto L1
L2:
(A) $1=40*i
(B) $2=4*j
(C) $3=a+$1
(D) $4=$3+$2
(E) *($4)=1
    j=j+1
L1:
    if j<n goto L2 else goto L3
L3:
       i=i+1
L4:
    if i<n goto L5 else goto L6
L6:
```

(a) *Identify the loop invarient computations from* $(A) \cdots (E)$. *Place them at appropriate places outside the loop. Do not perform any other transformation at this stage.*

(b) *Find the sequence of values in* `$1` *and* `$2`.

(c) *How do you modify the code for the computation of induction variables (a variable whose values forms an AP)* `$1` *and* `$2` *to reduce the cost of computation?*

(d) *Can the variable* `i, j` *be removed?*

(e) *How is the GCC x86-64 code?*

Ans.

(a) *(A) and (C) can be shifted below $L4$.*

```
L4:
(A) $1=40*i
(C) $3=a+$1
    if i<n goto L5 else goto L6
```

(b) *$\$1 : 0, 40, 80, \cdots$ and $\$2 : 0, 4, 8, \cdots$.*

(c)

```
    i=0
    $1=0
    goto L4
L5:
    j=0
    $2=0
    goto L1
L2:
(D) $4=$3+$2
(E) *($4)=1
    j=j+1
    $2=$2+4
L1:
    if j<n goto L2 else goto L3
L3:
    i=i+1
    $1=$1+40
L4:
    $3=a+$1
    if i<n goto L5 else goto L6
L6:
```

(d) *The purpose of* `i, j` *after the transformation is just to control the loop. This can be achieved using* `$1, $2` *where* `n` *is replaced by appropriate values.*
We compute two values at the beginning: `$5 = 4*n` *and* `$6=40*n`*. The inner loop continues as long as* `$2` $<$ `$5`*. Similarly, the outer loop continues as long as* `$1` $<$ `$6`*.*

```
    $1=0
    $5=4*n
    $6=40*n
    goto L4
L5:
    $2=0
    goto L1
L2:
(D) $4=$3+$2
(E) *($4)=1
    $2=$2+4
L1:
    if $2<$5 goto L2 else goto L3
```

```
L3:
    $1=$1+40
L4:
    $3=a+$1
    if $1<$6 goto L5 else goto L6
L6:
```

(e) *The code outline is*

```
    $1=4*n
    $2=a+$1     # $2=a+4n
    $3=44*n
    $4=a+$3     # $4=a+44n
    $5=-$1      # $5=-4n
    goto L3
L6:
    $2=$2+40
    if $2 == $4 goto L1
L3:
    $6=$2+$5    # $6=a+40i
L4:
    *$6=1       # *(a+40i+4j) = 1
    $6=$6+4     # $6=a+40i+4j
    if $6 <> $2 goto L4 else goto L6
L1:
```

The actual assembly code:

```
# 1st parameter in rdi
# 2nd parameter in esi/rsi
initMat:
.LFB0:
  testl  esi, esi          # if n <= 0
  jle  .L1                   #    goto .L1
  leal  -1(rsi), ecx       # ecx = n-1
  leaq  0(,rcx,4), rax     # rax = 4(n-1)
  leaq  4(rdi,rax), rdx   # rdx = a+4n
                             #  address of nth element of 0th row
  addq  rcx, rax           # rax = 5(n-1)
  leaq  (rcx,rax,2), rax  # rax = 11(n-1)
  leaq  44(rdi,rax,4), rsi # rsi = a+44n
                              #  address of the nth element of nth row
  notq  rcx                 # rcx = not(n-1)
                             #     = -n
  salq  $2, rcx             # rcx = -4n
  jmp  .L3                   # goto .L3
.L6:
  addq  $40, rdx            # rdx = rdx+40
                             #    address of the nth element of the
                             #    next row
```

```
                          #
  cmpq  rsi, rdx          # if rdx == a+44n
  je  .L1                   #    goto .L1
.L3:
  leaq  (rcx,rdx), rax    # rax = a+40i
.L4:
  movl  $1, (rax)           # a[i][j] = 1
  addq  $4, rax             # rax = a+40i+4(j+1)
  cmpq  rdx, rax          # if rax < rdx
  jne  .L4                  #    goto .L4, next in the row
  jmp  .L6                  # else goto .L6, next row
.L1:
  ret
```