



COSC 6365
Lecture 13
2008-02-26

CS@UH

Introduction to HPC

Lecture 13

Lennart Johnsson
Dept of Computer Science
Director TLC²



COSC 6365
Lecture 13
2008-02-26

CS@UH

Program transformations for vectorization - dependencies

- *True data dependencies*: read-after-write (RAW)
- *Anti-dependence*: write-after-read (WAR)
- *Output dependence*: write-after-write (WAW)
- *Loop-carried dependence*: a dependence of one of the above types occurs from one iteration to another



COSC 6365
Lecture 13
2008-02-26

CS@UH

Common program transformations for vectorization

1. Variable Renaming
2. Statement Reordering
3. Loop Distribution
4. Loop Reordering
5. Scalar Expansion
6. Variable Copying
7. Index Splitting
8. Node Splitting
9. Loop Unrolling
10. Loop Peeling
11. Loop Rerolling
12. Loop Collapsing



Loop Reordering

```
FOR I = 1 TO 100 DO  
  FOR J = 1 TO 100 DO  
     $X(I, J+1) = X(I, J) * Y(I, J)$   
  ENDFOR  
ENDFOR
```

Loop carried RAW dependence

```
FOR J = 1 TO 100 DO  
  FOR I = 1 TO 100 DO  
     $X(I, J+1) = X(I, J) * Y(I, J)$   
  ENDFOR  
ENDFOR
```

Inner loop now has no dependence

```
FOR J = 1 TO 100 DO  
   $X(1:100, J+1) = X(1:100, J) * Y(1:100, J)$   
ENDFOR
```



Matrix-Vector multiplication

$$y(i) = \sum_{j=1}^Q (A(i, j) \times x(j)), \quad \text{for all } i \in [1, P]$$

```
FOR I = 1 TO P DO
  Y(I) = 0
  FOR J = 1 TO Q DO
    Y(I) = Y(I)+A(I,J)*X(J)
  ENDFOR
ENDFOR
```

```
FOR I = 1 TO P DO
  Y(I) = 0
ENDFOR
FOR J = 1 TO Q DO
  FOR I = 1 TO P DO
    Y(I) = Y(I)+A(I,J)*X(J)
  ENDFOR
ENDFOR
```

```
Y(1:P) = 0
FOR J = 1 TO Q DO
  Y(1:P) = Y(1:P)+A(1:P,J)*X(J)
ENDFOR
```

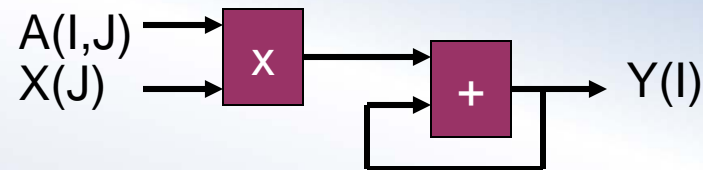
$$y(1:P) = \sum_{j=1}^Q (A(1:P, j) \times x(j)),$$



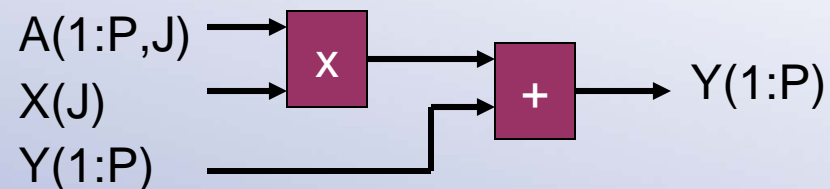
Matrix-Vector multiplication

$$y(i) = \sum_{j=1}^Q (A(i, j) \times x(j)), \quad \text{for all } i \in [1, P]$$

```
FOR I = 1 TO P DO
  Y(I) = 0
  FOR J = 1 TO Q DO
    Y(I) = Y(I)+A(I,J)*X(J)
  ENDFOR
ENDFOR
```



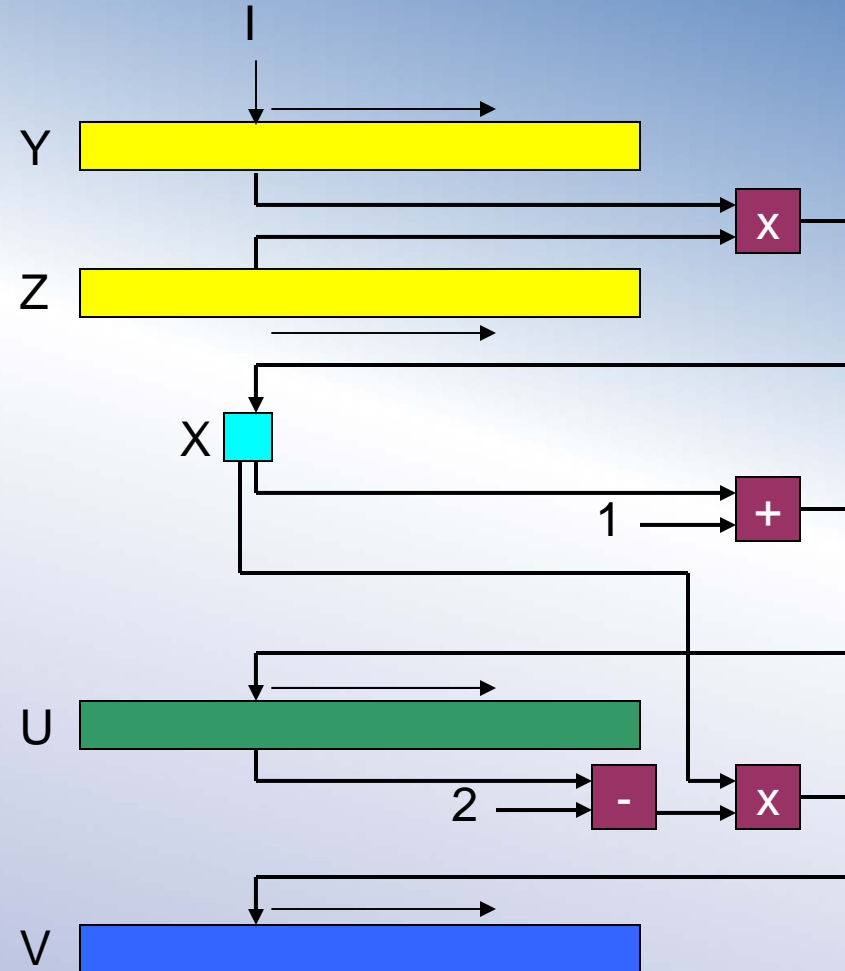
```
Y(1:P) = 0
FOR J = 1 TO Q DO
  Y(1:P) = Y(1:P)+A(1:P,J)*X(J)
ENDFOR
```





Scalar Expansion

```
FOR I = 1 TO N DO  
[1] X = Y(I)*Z(I)  
[2] U(I) = X+1  
[3] V(I) = X*(U(I)-2)  
ENDFOR
```





Scalar Expansion

```

FOR I = 1 TO N DO
[1] X = Y(I)*Z(I)
[2] U(I) = X+1
[3] V(I) = X*(U(I)-2)
ENDFOR

```

```

FOR I = 1 TO N DO
[1] TX(I) = Y(I)*Z(I)
[2] U(I) = TX(I)+1
[3] V(I) = TX(I)*(U(I)-2)
ENDFOR

```

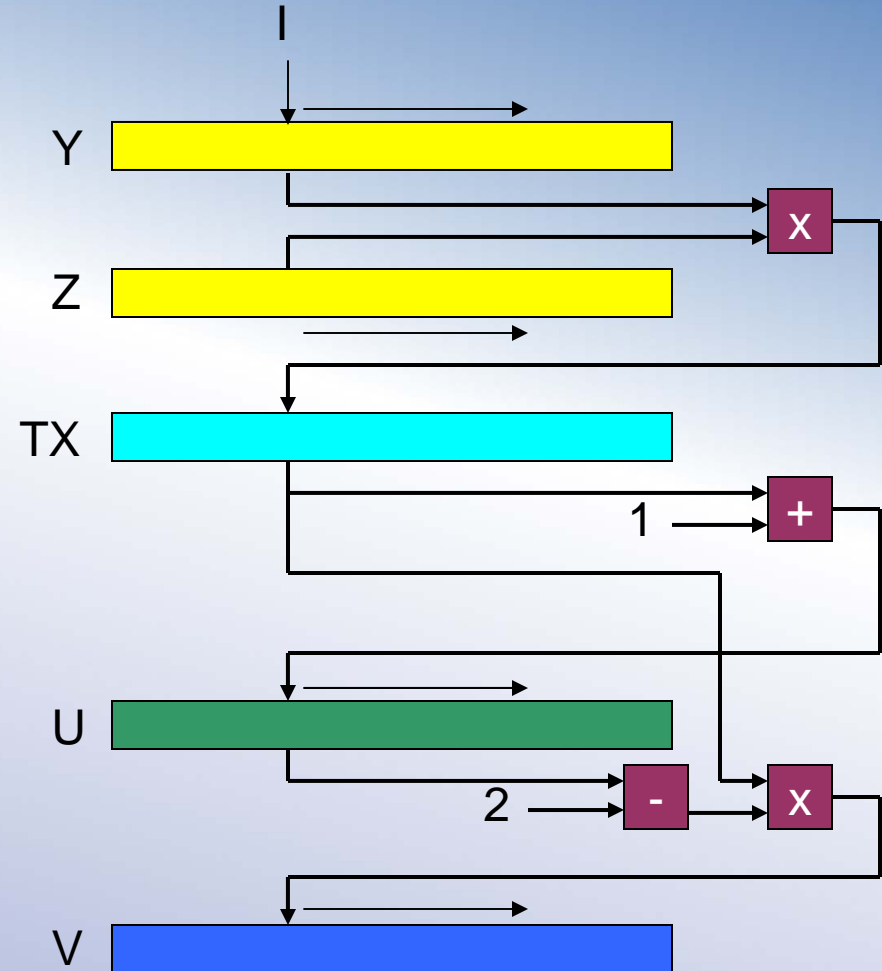
X is "promoted/expanded" from a scalar value to an array TX

The transformed loop can be vectorized

```

TX(1:100) = Y(1:100)*Z(1:100)
U(1:100) = TX(1:100)+1
V(1:100) = TX(1:100)*(U(1:100)-2)

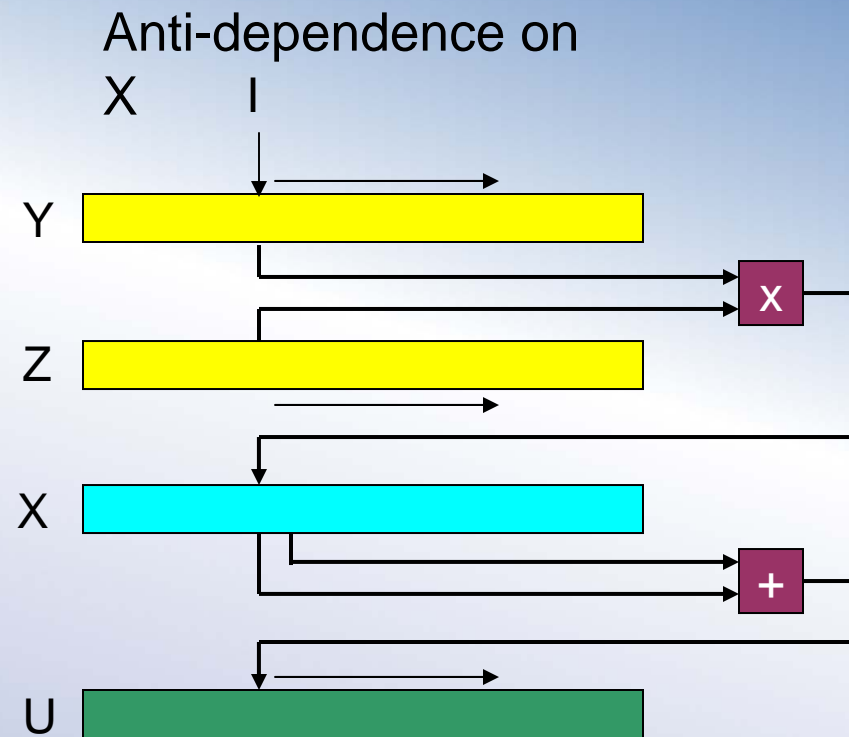
```





Variable Copy

```
FOR I = 1 TO N DO  
[1] X(I) = Y(I)*Z(I)  
[2] U(I) = X(I)+X(I+1)  
ENDFOR
```





Variable Copy

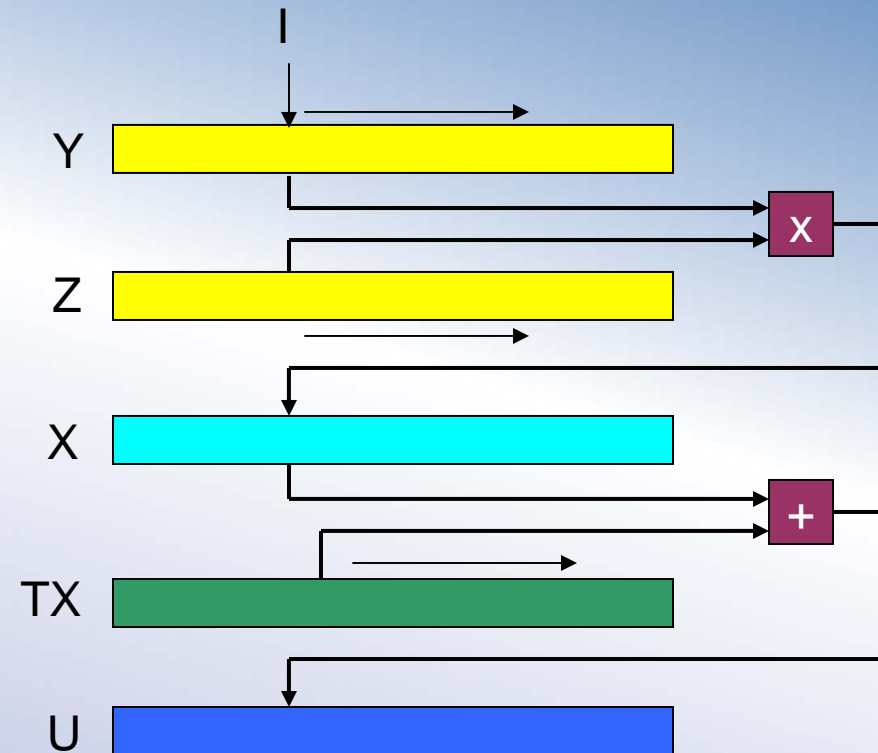
```
FOR I = 1 TO N DO
[1] X(I) = Y(I)*Z(I)
[2] U(I) = X(I)+X(I+1)
ENDFOR
```

TX(1:100) = X(2:101)

```
FOR I = 1 TO N DO
[1] X(I) = Y(I)*Z(I)
[2] U(I) = X(I)+TX(I)
ENDFOR
```

No anti-dependence

```
TX(1:100) = X(2:101)
X(1:100) = Y(1:100)*Z(1:100)
U(1:100) = X(1:100)+TX(1:100)
```

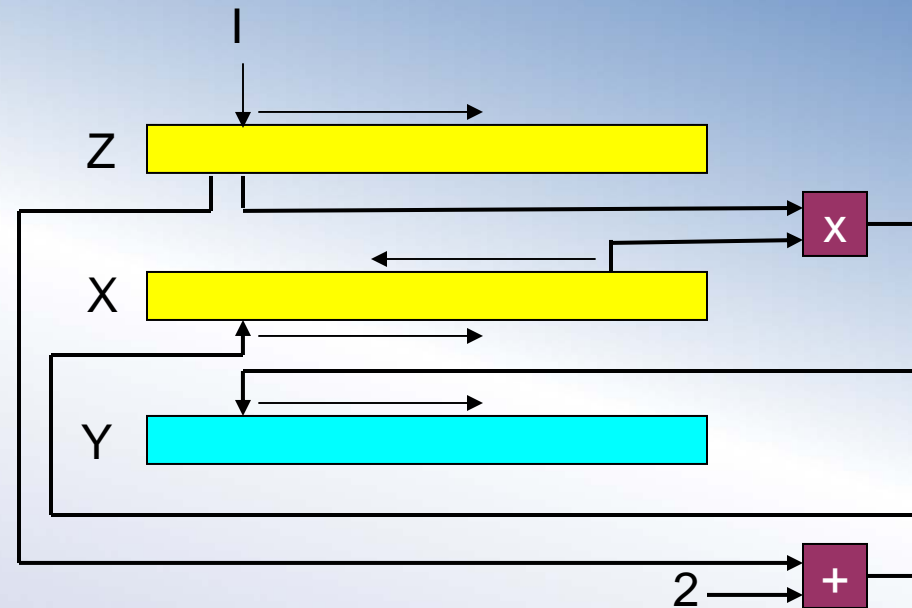


Vectorized



Index Splitting

```
FOR I = 1 TO 200 DO  
[1] Y(I) = X(201-I)+Z(I)  
[2] X(I) = Z(I-1)+2  
ENDFOR
```





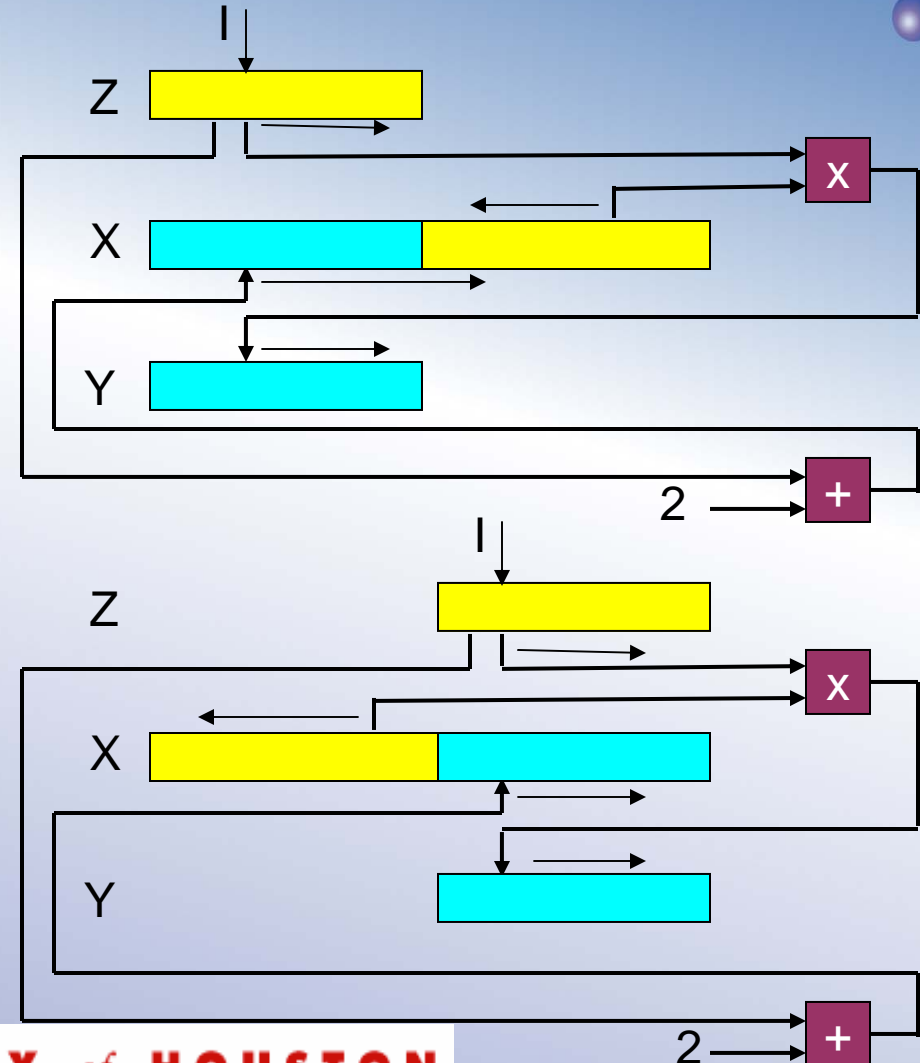
Index Splitting

```
FOR I = 1 TO 200 DO
[1] Y(I) = X(201-I)+Z(I)
[2] X(I) = Z(I-1)+2
ENDFOR
```

Split index

```
FOR I = 1 TO 100 DO
[1] Y(I) = X(201-I)+Z(I)
[2] X(I) = Z(I-1)+2
ENDFOR
```

```
FOR I = 101 TO 200 DO
[1] Y(I) = X(201-I)+Z(I)
[2] X(I) = Z(I-1)+2
ENDFOR
```





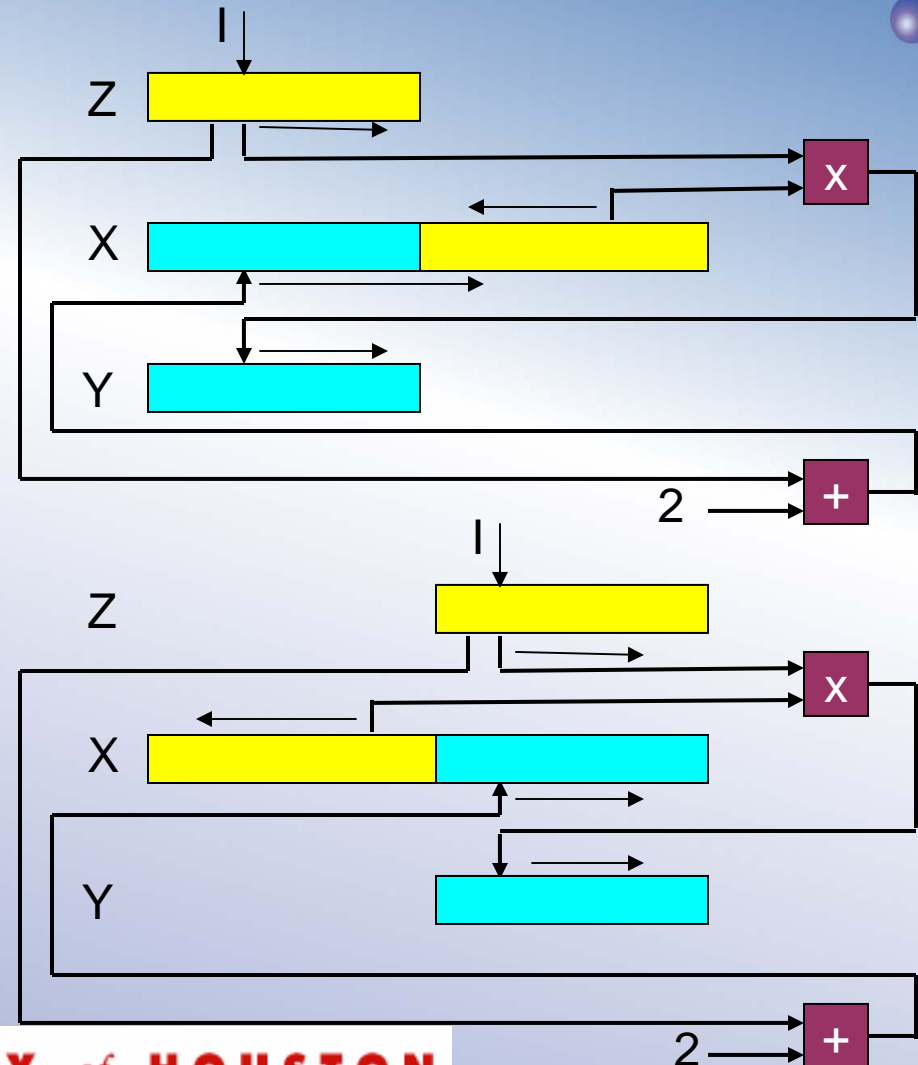
Index Splitting

```
FOR I = 1 TO 200 DO
[1] Y(I) = X(201-I)+Z(I)
[2] X(I) = Z(I-1)+2
ENDFOR
```

Vectorized

```
Y(1:100) = X(200:101:-1)+Z(1:100)
X(1:100) = Z(0:99)+2
```

```
Y(101:200) = X(100:1:-1)+Z(101:200)
X(101:200) = Z(100:199)+2
```





Node Splitting

```
FOR I = 1 TO 100 DO  
[1] Y(I) = X(I)+Z(I)*U(I)  
[2] X(I+1) = Y(I)*(U(I)-Z(I))  
ENDFOR
```

Has true dependence (RAW)
on Y and loop carried on X

Node splitting

```
FOR I = 1 TO 100 DO  
    T1(I) = Z(I)*U(I)  
    T2(I) = U(I)-Z(I)  
[1] Y(I) = X(I)+T1(I)  
[2] X(I+1) = Y(I)*T2(I)  
ENDFOR
```

Partial vectorization

```
T1(1:100) = Z(1:100)*U(1:100)  
T2(1:100) = U(1:100)-Z(1:100)  
FOR I = 1 TO 100 DO  
[1] Y(I) = X(I)+T1(I)  
[2] X(I+1) = Y(I)*T2(I)  
ENDFOR
```



COSC 6365
Lecture 13
2008-02-26

CS@UH

Loop Unrolling

```
FOR I = 1 TO N DO  
    SUM = SUM+X(I)  
ENDFOR
```

The loop unrolled to depth 4

```
FOR I = 1 TO N-3 STEP 4 DO  
    SUM = SUM+X(I)+X(I+1)+X(I+2)+X(I+3)  
ENDFOR
```

```
FOR J = I TO N DO  
    SUM = SUM+X(J)  
ENDFOR
```

Code for handling the remainder
when N isn't a multiple of 4.



Loop Unrolling

Vectorized and depth 6 unrolled MM

Vectorizable MM

```

FOR J = 1 TO R DO
  FOR I = 1 TO P DO
    A(I,J) = 0.
  ENDFOR
ENDFOR

```

Outer Product MM

```

FOR K = 1 TO Q DO
  FOR J = 1 TO R DO
    FOR I = 1 TO P DO
      A(I,J) = A(I,J)+B(I,K)*C(K,J)
    ENDFOR
  ENDFOR
ENDFOR

```

```

FOR J = 1 TO R DO
  A(1:P,J) = 0.
ENDFOR

```

```

FOR K = 1 TO Q-5 STEP 6 DO
  FOR J = 1 TO R DO

```

```

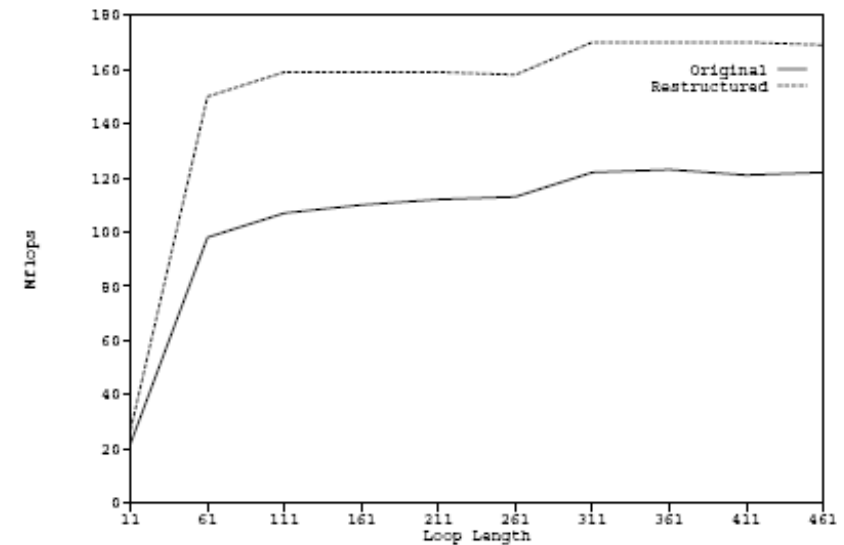
    A(1:P,J) = A(1:P,J)+B(1:P,K)*C(K,J)+B(1:P,K+1)*C(K+1,J)+
    B(1:P,K+2,J)*C(K+2,J)+B(1:P,K+3)*C(K+3,J)+
    B(1:P,K+4)*C(K+4,J)+B(1:P,K+5)*C(K+5,J)

```

```

  ENDFOR
ENDFOR

```





Loop Unrolling

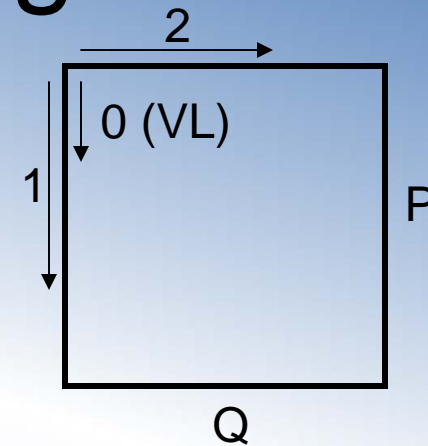
Matrix-Vector Multiplication, vectorized

$$Y(1:P) = 0.$$

FOR J = 1 TO Q DO

$$Y(1:P) = Y(1:P) + A(1:P, J) * X(J)$$

ENDFOR



Matrix-Vector Multiplication, unrolled and vectorized

$$Y(1:P) = 0.$$

FOR J = 1 TO Q-3 STEP 4 DO

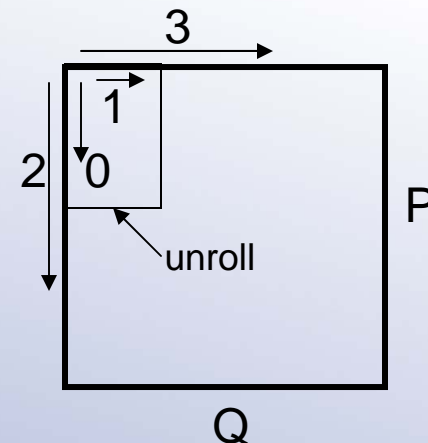
$$Y(1:P) = Y(1:P) + A(1:P, J) * X(J) + A(1:P, J+1) * X(J+1) + A(1:P, J+2) * X(J+2) + A(1:P, J+3) * X(J+3)$$

ENDFOR

FOR J = 1 TO Q DO

$$Y(1:P) = Y(1:P) + A(1:P, J) * X(J)$$

ENDFOR



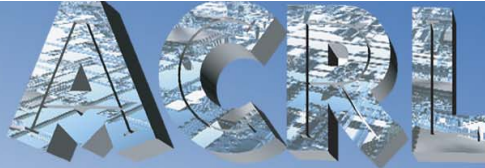


Loop Unrolling

Matrix-Vector Multiplication, vectorized

Operation	Cycles
Load A(:,J)	12+VL
Multiply A(:,J)*X(J)	7 (chained with 1 st load)
Load Y(:)	12+VL
Add A(:,J)*X(J) to Y(:)	6 (chained with 2 nd load)
Store Y(:)	12+VL
Stall for store	4 (store depend on add)
Total	46+3VL

$$T_N = 10 + \left\lceil \frac{N}{64} \right\rceil (15 + 46) + 3N.$$



ADVANCED COMPUTING RESEARCH LABORATORY

COSC 6365
Lecture 13
2008-02-26



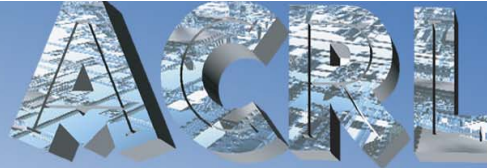
Loop Unrolling



Operation	Cycles
Load Y(:)	12+VL
Load A(:,J)	12+VL
Multiply A(:,J)*X(J)	7 (chained with load)
Add Y(:)+A(:,J)*X(J)	6 (chained with load)
Load A(:,J+1)	12+VL
Multiply A(:,J+1)*X(J+1)	7 (chained with load)
Add Y(:)+A(:,J+1)*X(J+1)	6 (chained with load)
Stall for load-mult-add chain	4
Load A(:,J+1)	12+VL
Multiply A(:,J+1)*X(J+1)	7 (chained with load)
Add Y(:)+A(:,J+1)*X(J+1)	6 (chained with load)
Stall for load-mult-add chain	4
Load A(:,J+1)	12+VL
Multiply A(:,J+1)*X(J+1)	7 (chained with load)
Add Y(:)+A(:,J+1)*X(J+1)	6 (chained with load)
Stall for load-mult-add chain	4
Store Y(:)	12+VL
Stall for load-mult-add chain	4 (store depend on add)
Total	140+6VL

Matrix-Vector
Multiplication,
unrolled and
vectorized

$$T_N = 10 + \left\lceil \frac{N/4}{64} \right\rceil (15 + 104) + 6(N/4)$$



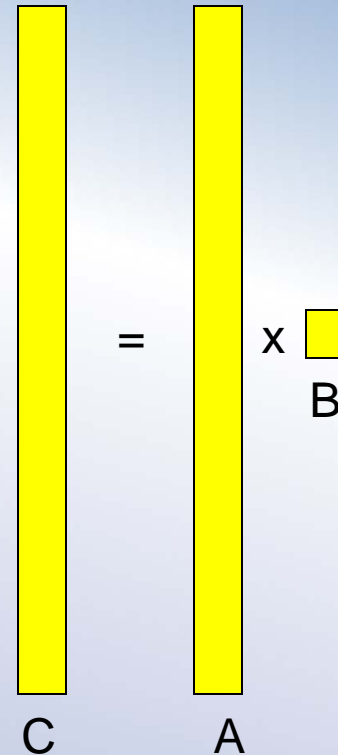
ADVANCED COMPUTING RESEARCH LABORATORY

COSC 6365
Lecture 13
2008-02-26

CS@UH

Loop Unrolling

```
FOR I = 1 TO P DO  
  FOR J = 1 TO 4 DO  
    C(I,J) = 0.  
    FOR K = 1 TO 4 DO  
      C(I,J) = C(I,J)+A(I,K)*B(K,J)  
    ENDFOR  
  ENDFOR  
ENDFOR
```





Loop Unrolling

```
FOR I = 1 TO P DO
  FOR J = 1 TO 4 DO
    C(I,J) = 0.
    FOR K = 1 TO 4 DO
      C(I,J) = C(I,J)+A(I,K)*B(K,J)
    ENDFOR
  ENDFOR
ENDFOR
```

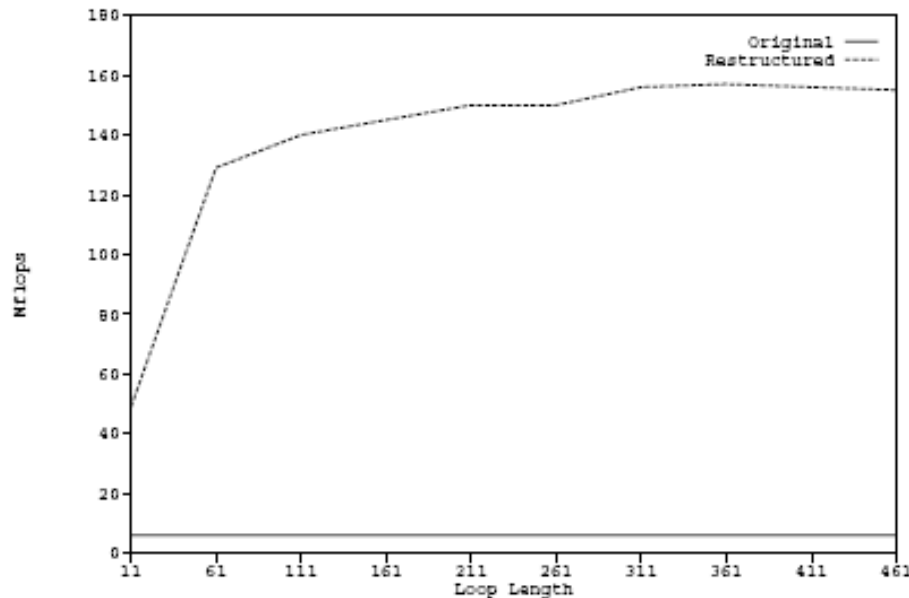
Unrolling K and J loops

```
FOR J = 1 TO 4 DO
  C(1:P,J) = 0.
ENDFOR
```

$$C(1:P,1) = A(1:P,1)*B(1,1)+A(1:P,2)*B(2,1)+A(1:P,3)*B(3,1)+A(1:P,4)*B(4,1)$$

$$C(1:P,2) = A(1:P,1)*B(1,2)+A(1:P,2)*B(2,2)+A(1:P,3)*B(3,2)+A(1:P,4)*B(4,2)$$

$$C(1:P,3) = A(1:P,1)*B(1,3)+A(1:P,2)*B(2,3)+A(1:P,3)*B(3,3)+A(1:P,4)*B(4,3)$$

$$C(1:P,4) = A(1:P,1)*B(1,4)+A(1:P,2)*B(2,4)+A(1:P,3)*B(3,4)+A(1:P,4)*B(4,4)$$




COSC 6365
Lecture 13
2008-02-26

CS@UH

Loop Collapsing

```
FOR J = 1 TO R DO  
  FOR I = 1 TO P DO  
    A(I,J) = 0.  
  ENDFOR  
ENDFOR
```

Collapsed Loops

```
A(1:PR) = 0.
```



Loop Peeling

```

FOR J = 1 TO R DO
  FOR I = 1 TO P DO
S1    X(I,J) = Y(I,J)+3*Y(I-1,J)
      IF J<2 THEN GOTO S4
      IF J<R THEN GOTO S3
S2    Y(I,J) = X(I,J)
S3    Y(I-1,J) = Z(I,J)
S4    Z(I,J) = X(I,J)
      ENDFOR
  ENDFOR

```

Conditionals → vectorization hard

```

J=1
FOR I = 1 TO P DO
S1    X(I,1) = Y(I,1)+3*Y(I-1,1)
S4    Z(I,1) = X(I,1)
  ENDFOR

```

```

FOR J = 2 TO R-1 DO
  FOR I = 1 TO P DO
S1    X(I,J) = Y(I,J)+3*Y(I-1,J)
S3    Y(I-1,J) = Z(I,J)
S4    Z(I,J) = X(I,J)
  ENDFOR
  ENDFOR

```

```

J=R
FOR I = 1 TO P DO
S1    X(I,R) = Y(I,R)+3*Y(I-1,R)
S2    Y(I,R) = X(I,R)
S3    Y(I-1,R) = Z(I,R)
S4    Z(I,R) = X(I,R)
  ENDFOR

```



Loop Peeling

J=1

```
FOR I = 1 TO P DO
S1  X(I,1) = Y(I,1)+3*Y(I-1,1)
S4  Z(I,1) = X(I,1)
ENDFOR
```

```
X(1:P,1) = Y(1:P,1)+3*Y(0:P-1,1)
Z(1:P,1) = X(1:P,1)
```

```
FOR J = 2 TO R-1 DO
  FOR I = 1 TO P DO
```

```
S1  X(I,J) = Y(I,J)+3*Y(I-1,J)
S3  Y(I-1,J) = Z(I,J)
S4  Z(I,J) = X(I,J)
  ENDFOR
ENDFOR
```

```
X(1:P,2:R-1) = Y(1:P,2:R-1)+3*Y(0:P-1,2:R-1)
Y(0:P-1,2:R-1) = Z(1:P,2:R-1)
Z(1:P,2:R-1) = X(1:P,2:R-1)
```

J=R

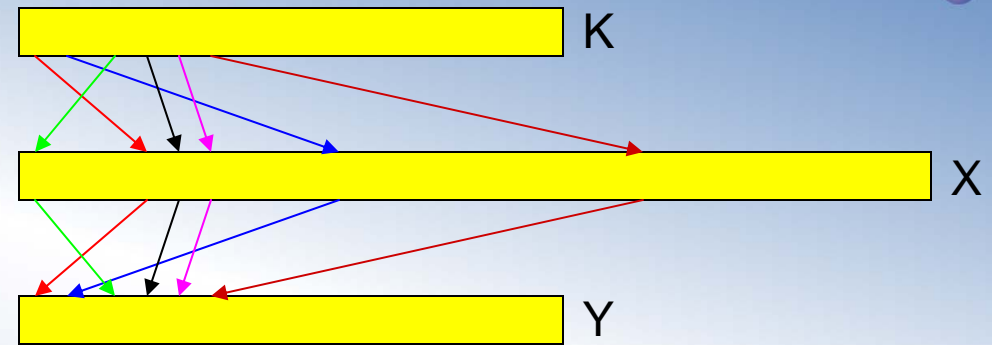
```
FOR I = 1 TO P DO
S1  X(I,R) = Y(I,R)+3*Y(I-1,R)
S2  Y(I,R) = X(I,R)
S3  Y(I-1,R) = Z(I,R)
S4  Z(I,R) = X(I,R)
ENDFOR
```

```
FOR I = 1 TO P DO
S1  X(I,R) = Y(I,R)+3*Y(I-1,R)
S2  Y(I,R) = X(I,R)
ENDFOR
Y(0:P-1,R) = Z(1:P,R)
Z(1:P,R) = X(1:P,R)
```

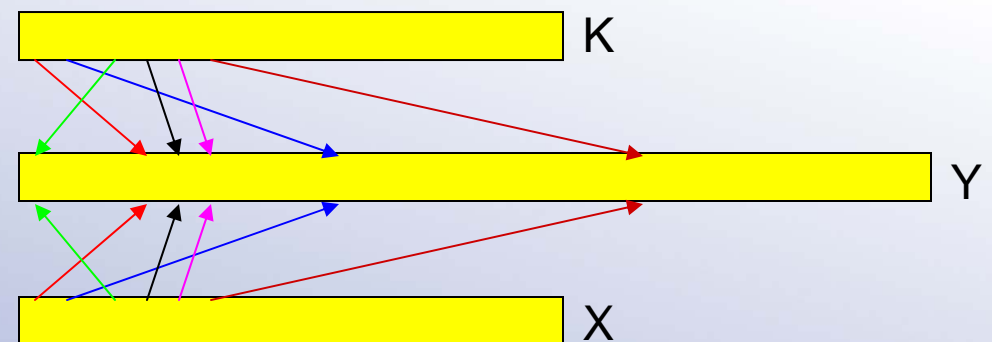


Indirect Addressing

Gather: $Y(I) = X(K(I))$



Scatter: $Y(K(I)) = X(I)$





Indirect Addressing

```
FOR I = 1 TO N DO
  A(IA(I)) = B(IB(I))+C(IC(I))
ENDFOR
```

VP-1

VP-3

Operation	Cycles	Operation	Cycles
Load IB into V0	12+VL	Load IB into V0 and IC into V1	12+VL
Load IC into V1	12+VL	Load IA into V5	12+VL
Load B indirect into V2 using V0	12+VL	Load B indirect into V2 using V0	12+VL
Load C indirect into V3 using V1	12+VL	and Load C indirect into V3 using V1	
Add B and C	6+VL	Add B and C chained with load	6
Stall (add must wait for load)	4	Store A indirect using V5	12
Load IA into V5	12+VL	Total	54+3VL
Store A indirect using V5	12+VL		
Total	82+7VL		

$$T_N = 10 + \left\lceil \frac{N}{64} \right\rceil (15 + 82) + 7N$$

$$T_N = 10 + \left\lceil \frac{N}{64} \right\rceil (15 + 54) + 3N.$$



Vectorizing Conditionals

```

FOR I = 1 TO N DO
  IF (B(I).NE.) THEN
    A(I) = A(I)/B(I)
  ENDIF
ENDFOR

```

Mask mode

	Operation	Cycles
Load A into V1	Load A	12+VL
Load B into V2	Load B	12+VL
Set F0 to 0.0	Set F0 to 0.0	1
Set VMR to 1 if V2(i).NE.F0, 0 otherwise	Set VMR	VL
Divide V1 by V2 under mask	Divide V1 by V2 under VMR	20+VL
Set the VMR to all 1's	Set the VMR to all 1's	1
Store the result in A	Store the result	12+VL
	<u>Stall (store must wait for divide)</u>	<u>4</u>
	Total	62+5VL



Vectorizing Conditionals

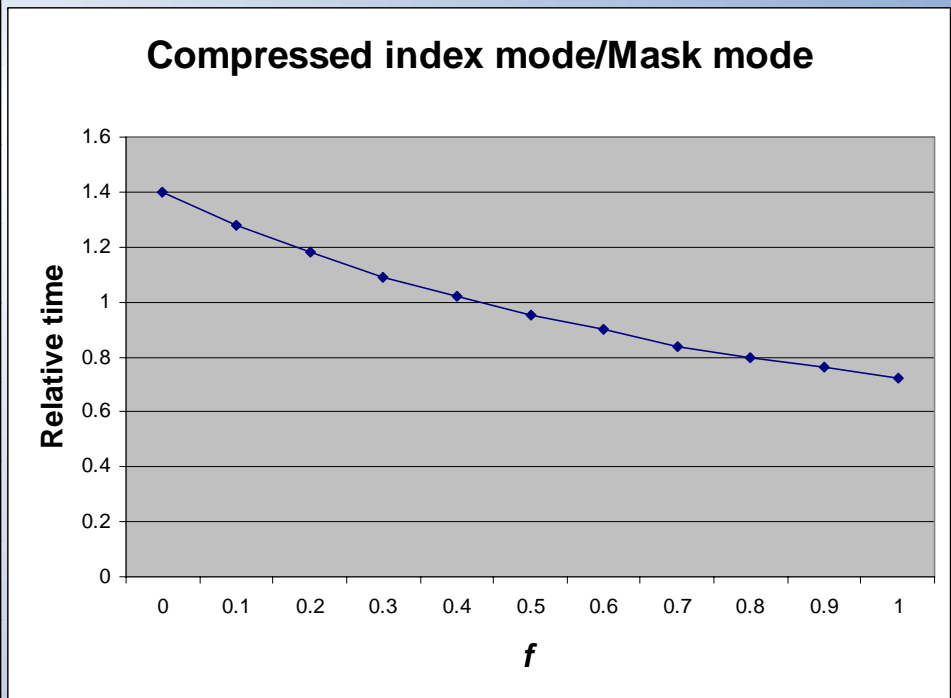
Compressed index mode

	Operation	Cycles
Load B into V1	Load A	12+VL
Set F0 to 0.0	Set F0 to 0.0	1
Set VMR to 1 if V1(i).ne.0, 0 otherwise	Set VMR	VL
Create index vector in V2 under mask	Create Index vector	VL
Count non-zeroes in VMR and set VLR	Count non-zeroes and set VLR	1
Set VMR to all 1's	Set the VMR to all 1's	1
Load A into V3 indirect using V2	Load A indirect	12+fVL
Load B into V4 indirect using V2	Load B indirect	12+fVL
Divide V3 by V4	Divide A by B	20+fVL
Store the result indirect using V2	Stall (divide must wait for load)	4
	Store the result indirect	12+fVL
	Stall (store must wait for divide)	4
	<u>Total</u>	<u>79+(3+4fVL)</u>



Vectorizing Conditionals

f	Mask	Compressed index	Compr./Mask
0.0	0.168	0.234	1.40
0.1	0.168	0.214	1.28
0.2	0.168	0.197	1.18
0.3	0.168	0.183	1.09
0.4	0.168	0.170	1.02
0.5	0.168	0.160	0.95
0.6	0.168	0.150	0.90
0.7	0.168	0.142	0.84
0.8	0.168	0.134	0.80
0.9	0.168	0.127	0.76
1.0	0.168	0.121	0.72





COSC 6365
Lecture 13
2008-02-26

CS@UH

References

- John M Levesque and Joel W. Williamson. *A Guidebook to Fortran on Supercomputers*, Academic Press, 1989.
- Hans Zima and Barbara Chapman. *Supercompilers for Parallel and Vector Computers*, ACM Press, 1991.