



COSC 6365  
Lecture 7  
2008-02-05

**CS@UH**

# Introduction to HPC

## Lecture 7

Lennart Johnsson  
Dept of Computer Science  
Director TLC<sup>2</sup>



COSC 6365  
Lecture 7  
2008-02-05

**CS@UH**

# Matrix operations and Memory operations

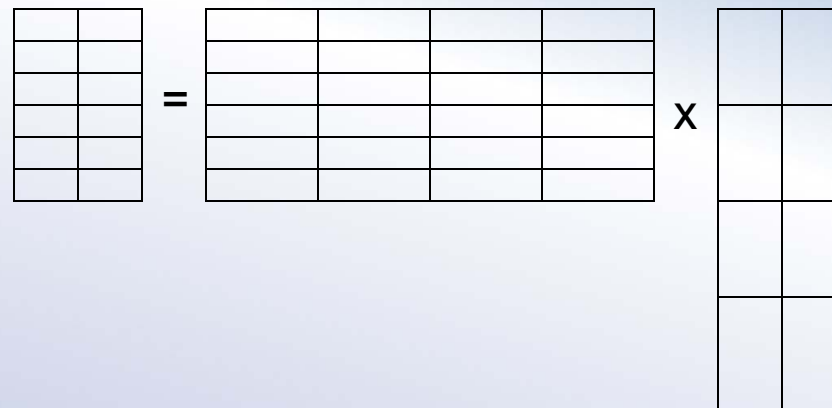
- Impact of blocking on memory accesses
- Strides and memory system performance



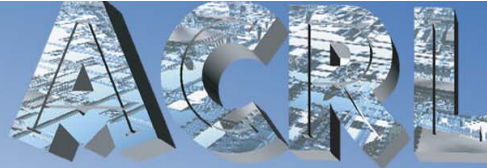
COSC 6365  
Lecture 7  
2008-02-05

**CS@UH**

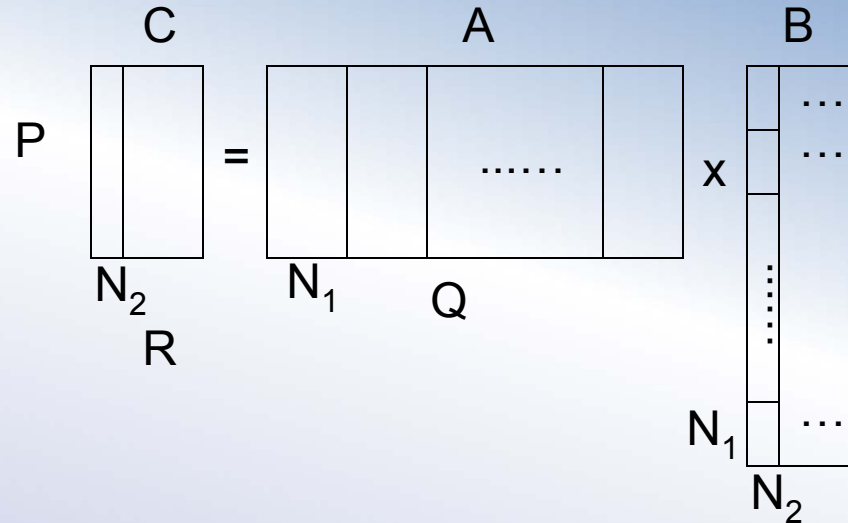
# Matrix-Matrix Multiplication



Blocking



# Matrix multiplication Maximize reuse of B - columns first



Loop on R ...  
 Loop on Q ..  
 Loop on R: 1 – N<sub>2</sub>  
 Loop on Q: 1 – N<sub>1</sub>  
 Loop on P

Reuse of B: Load N<sub>1</sub> elements of N<sub>2</sub> columns:  $K = N_1 \times N_2$   
 Loads: A:  $(P \times N_1)(Q/N_1)(R/N_2)$ , B:  $(Q \times R)$ , C:  $(P \times N_2)(Q/N_1 - 1)(R/N_2) =$

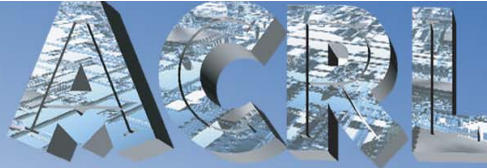
$$= (PQR/N_2) + (PQR/N_1) + QR - PR$$

Stores: C:  $(P \times N_2)(Q/N_1)(R/N_2) = (PQR)/N_1$

Minimize memory load/stores:  $PQR(2N_2/K + 1/N_2)$

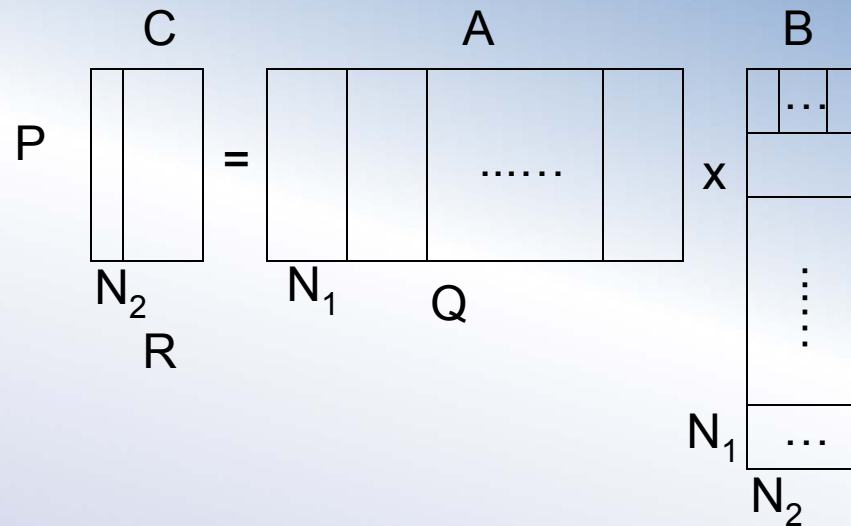
$$N_2 = \sqrt{K/2} \quad N_1 = \sqrt{2K}$$

Memory Operations:  $PQR2\sqrt{2/K} + (Q-P)R$



# Matrix multiplication

## Maximize reuse of B - rows first



Loop on Q ...  
 Loop on R ..  
 Loop on R: 1 - N<sub>2</sub>  
 Loop on Q: 1 - N<sub>1</sub>  
 Loop on P

Reuse of B: Load N<sub>1</sub> elements of N<sub>2</sub> columns:  $K = N_1 \times N_2$   
 Loads: A:  $(P \times N_1)(R/N_2)(Q/N_1)$ , B:  $(Q \times R)$ , C:  $(P \times N_2)(R/N_2)(Q/N_1 - 1) =$   
 $= (PRQ/N_2) + (PRQ/N_1) + QR - PR$

Stores: C:  $(P \times N_2)(R/N_2)(Q/N_1) = (PRQ)/N_1$

Minimize memory load/stores:  $PRQ(2N_2/K + 1/N_2)$

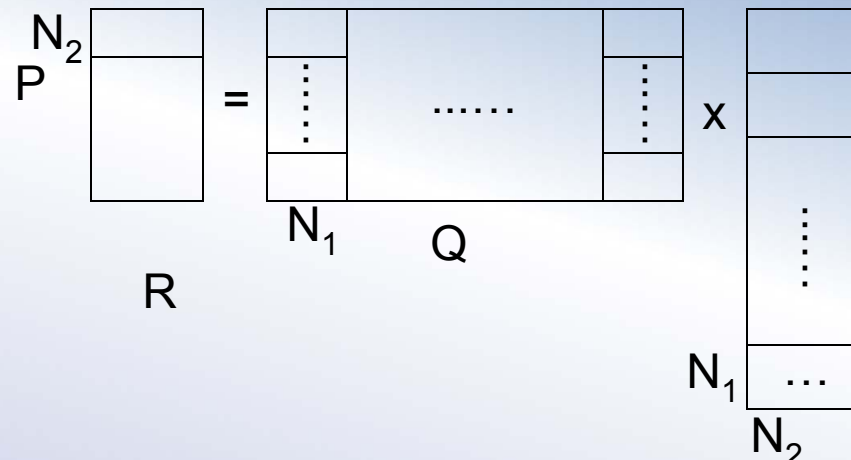
$N_2 = \sqrt{(K/2)}$       $N_1 = \sqrt{2K}$

Memory Operations:  $PRQ2\sqrt{(2/K)} + (Q-P)R$



# Matrix multiplication

## Maximize reuse of A - columns first



Loop on Q  
 Loop on P ...  
 Loop on R ..  
 Loop on Q: 1 -  $N_1$   
 Loop on P: 1 -  $N_2$

Reuse of B: Load  $N_1$  elements of  $N_2$  columns:  $K = N_1 \times N_2$   
 Loads: A:  $PQ$ , B:  $(N_1 \times R)(P/N_2)(Q/N_1)$ , C:  $(N_2 \times R)(P/N_2)(Q/N_1 - 1) =$   
 $= (RPQ/N_2) + (RPQ/N_1) + PQ - RP$

Stores: C:  $(N_2 \times R)(P/N_2)(Q/N_1) = (RPQ)/N_1$

Minimize memory load/stores:  $RPQ(2N_2/K + 1/N_2)$

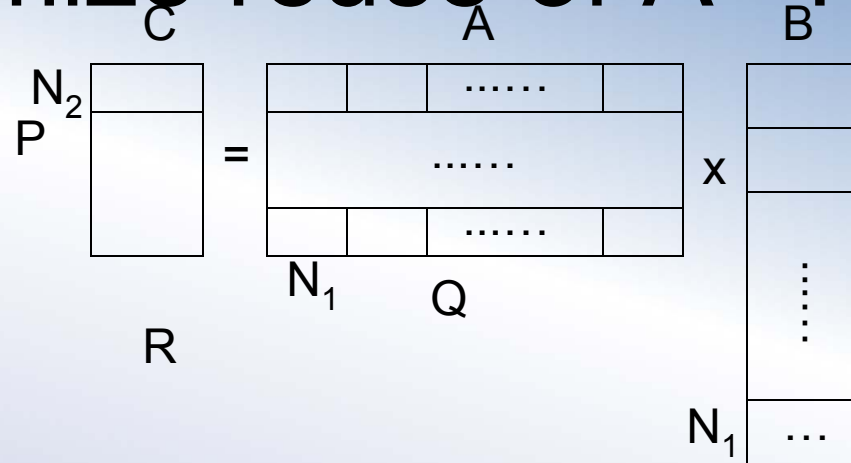
$$N_2 = \sqrt{K/2} \quad N_1 = \sqrt{2K}$$

Memory Operations:  $RPQ2\sqrt{(2/K)} + (Q-R)P$



# Matrix multiplication

## Maximize reuse of A - rows first



Loop on P  
 Loop on Q ...  
 Loop on R ..  
 Loop on Q: 1 -  $N_1$   
 Loop on P: 1 -  $N_2$

Reuse of B: Load  $N_1$  elements of  $N_2$  columns:  $K = N_1 \times N_2$   
 Loads: A:  $PQ$ , B:  $(N_1 \times R)(Q/N_1)(P/N_2)$ , C:  $(N_2 \times R)(Q/N_1 - 1)(P/N_2) =$   
 $= (RQP/N_2) + (RQP/N_1) + PQ - RP$

Stores: C:  $(N_2 \times R)(Q/N_1)(P/N_2) = (RQP)/N_1$

Minimize memory load/stores:  $RQP(2N_2/K + 1/N_2)$

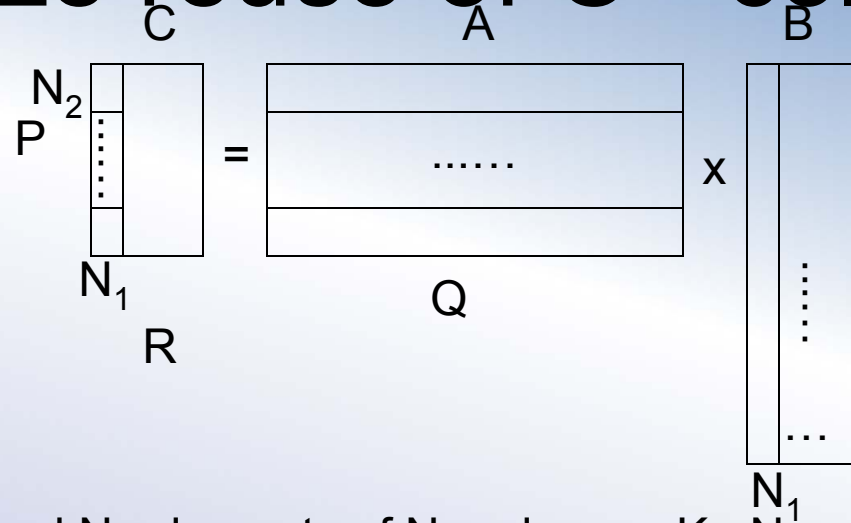
$N_2 = \sqrt{K/2}$       $N_1 = \sqrt{2K}$

Memory Operations:  $RQP2\sqrt{2/K} + (Q-R)P$



# Matrix multiplication

## Maximize reuse of C - columns first



Loop on R  
 Loop on P ...  
 Loop on R: 1 – N<sub>1</sub>  
 Loop on Q  
 Loop on P: 1 – N<sub>2</sub>

Reuse of B: Load N<sub>1</sub> elements of N<sub>2</sub> columns:  $K = N_1 \times N_2$

Loads: A:  $(N_2 \times Q)(P/N_2)(R/N_1)$ , B:  $(Q \times N_1)(P/N_2)(R/N_1) =$   
 $= (QPR/N_1) + (QPR/N_2)$

Stores: C:  $(N_2 \times N_1)(P/N_2)(R/N_1) = RP$

Minimize memory load/stores:  $QPR(N_2/K + 1/N_2)$

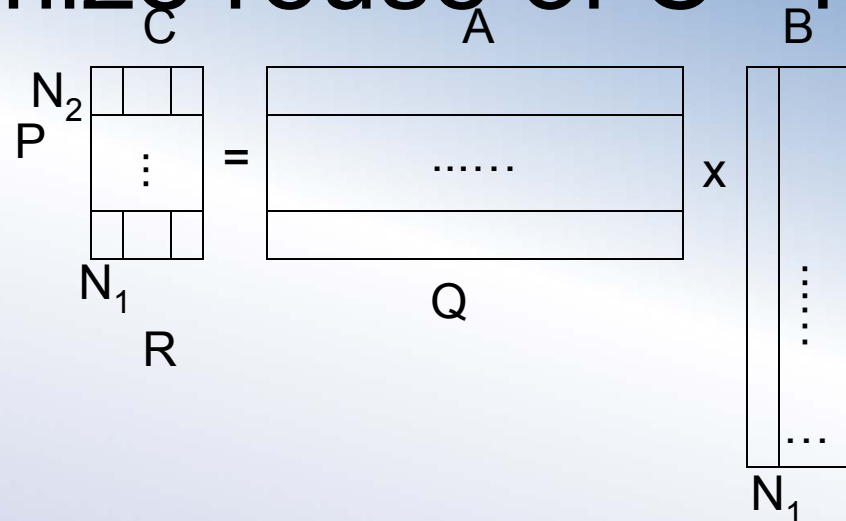
$N_2 = \sqrt{K}$        $N_1 = \sqrt{K}$

Memory Operations:  $QPR/2\sqrt{K} + RP$



# Matrix multiplication

## Maximize reuse of C - rows first



Loop on P  
 Loop on R ...  
 Loop on R 1 -  $N_1$   
 Loop on Q  
 Loop on P: 1 -  $N_2$

Reuse of B: Load  $N_1$  elements of  $N_2$  columns:  $K = N_1 \times N_2$

Loads: A:  $(N_2 \times Q)(R/N_1)(P/N_2)$ , B:  $(Q \times N_1)(R/N_1)(P/N_2) =$

$$= (QRP/N_1) + (QRP/N_2)$$

Stores: C:  $(N_2 \times N_1)(R/N_1)(P/N_2) = RP$

Minimize memory load/stores:  $QRP(N_2/K + 1/N_2)$

$$N_2 = \sqrt{K} \quad N_1 = \sqrt{K}$$

Memory Operations:  $QRP2/\sqrt{K} + RP$



COSC 6365  
Lecture 7  
2008-02-05

**CS@UH**

# Memory Operations

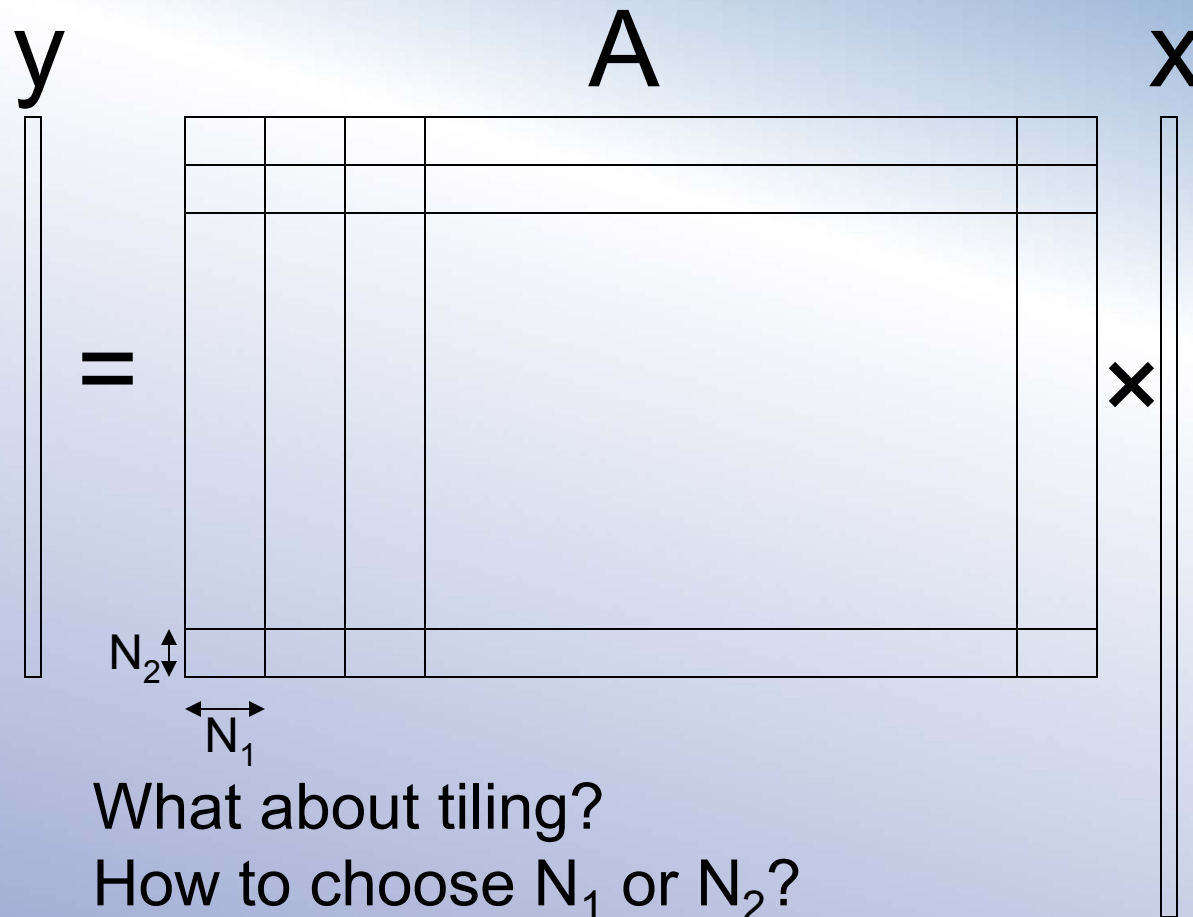
Reuse of B, columns first:  $PQR2\sqrt{(2/K)} + (Q-P)R$   
rows first:  $PRQ2\sqrt{(2/K)} + (Q-P)R$

Reuse of A, columns first:  $RPQ2\sqrt{(2/K)} + (Q-R)P$   
rows first:  $RQP2\sqrt{(2/K)} + (Q-R)P$

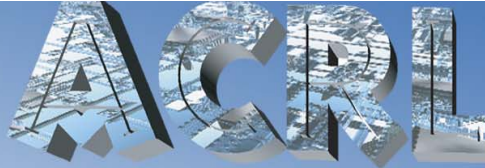
Reuse of C, columns first:  $QPR2/\sqrt{K} + RP$   
rows first:  $QRP2/\sqrt{K} + RP$



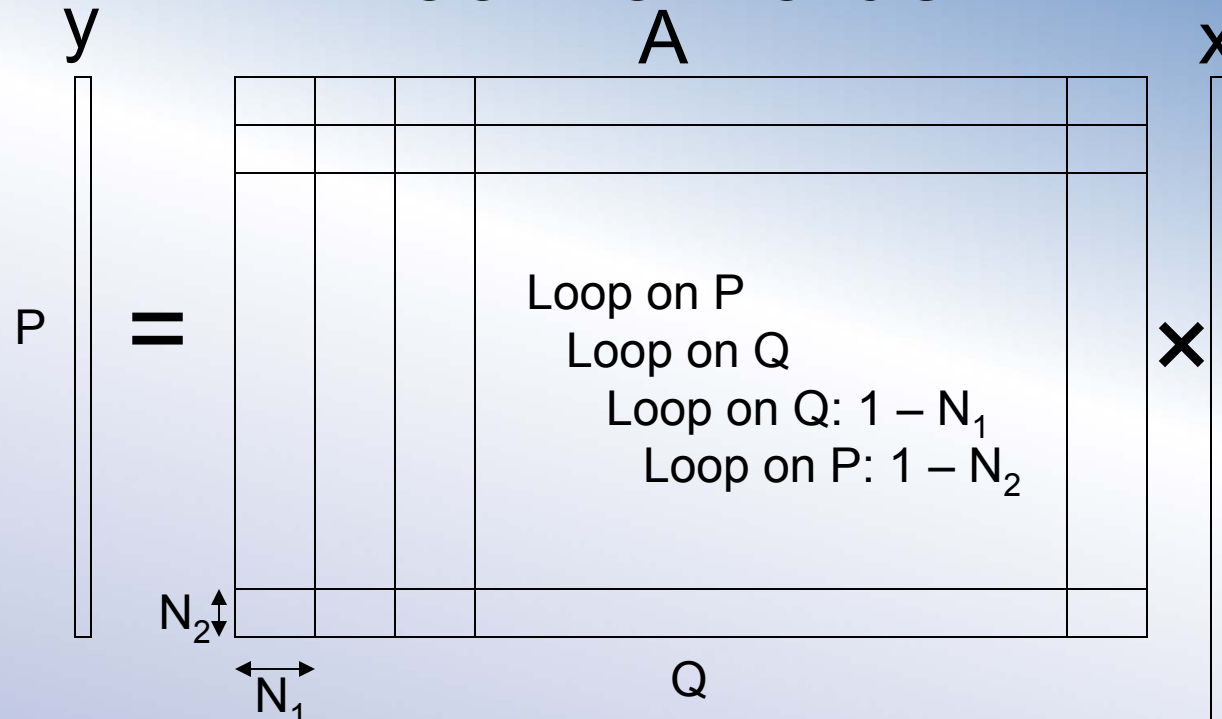
# Recall - Performance Workload - Memory



What about tiling?  
How to choose  $N_1$  or  $N_2$ ?



# Matrix-vector multiplication Block row order



Loads: A:  $(N_1 \times N_2)(Q/N_1)(P/N_2)$ , X:  $N_1(Q/N_1)(P/N_2)$

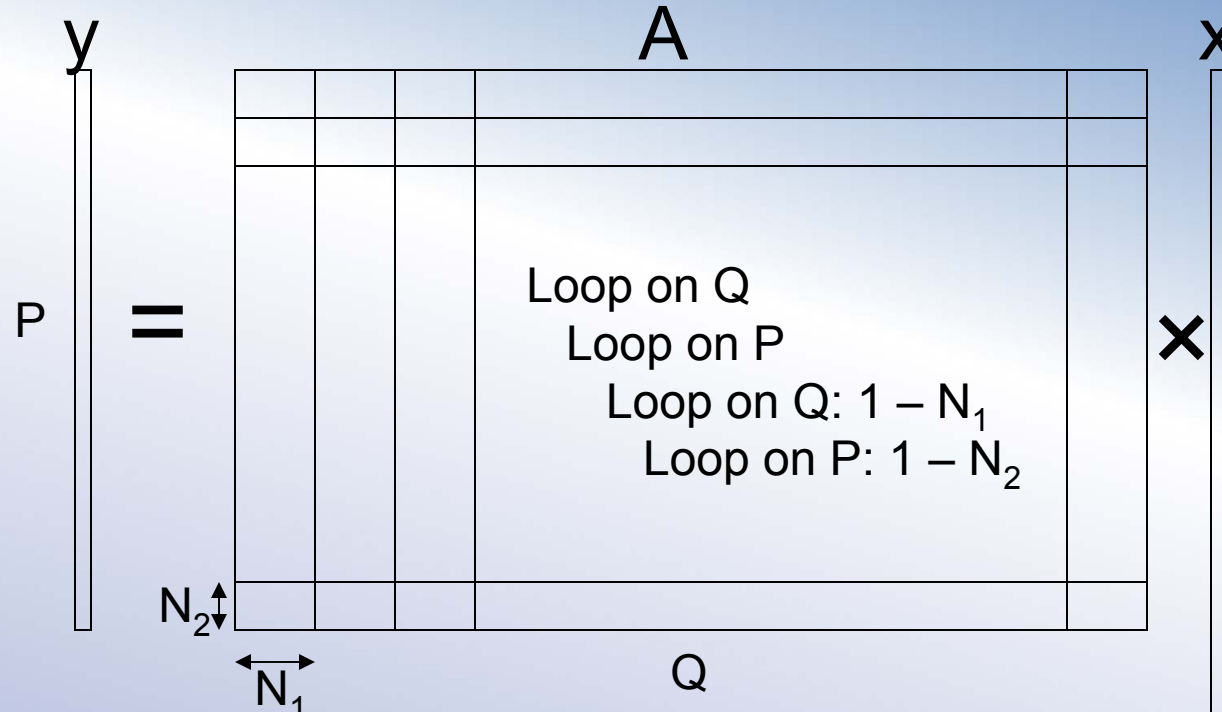
Stores: Y:  $N_2(P/N_2)$

Memory operations:  $QP + QP/N_2 + P$

Minimize memory operations for  $N_1 + N_2 = K \implies N_2 = K - 1$



# Matrix-vector multiplication Block column order



Loads:  $A: (N_1 \times N_2)(P/N_2)(Q/N_1)$ ,  $X: N_1(Q/N_1)$ ,  $Y: N_2(P/N_2)(Q/N_1 - 1)$

Stores:  $Y: N_2(P/N_2)(Q/N_1)$

Memory operations:  $PQ + 2PQ/N_1 + Q - P$

Minimize memory operations for  $N_1 + N_2 = K \implies N_1 = K - 1$



COSC 6365  
Lecture 7  
2008-02-05

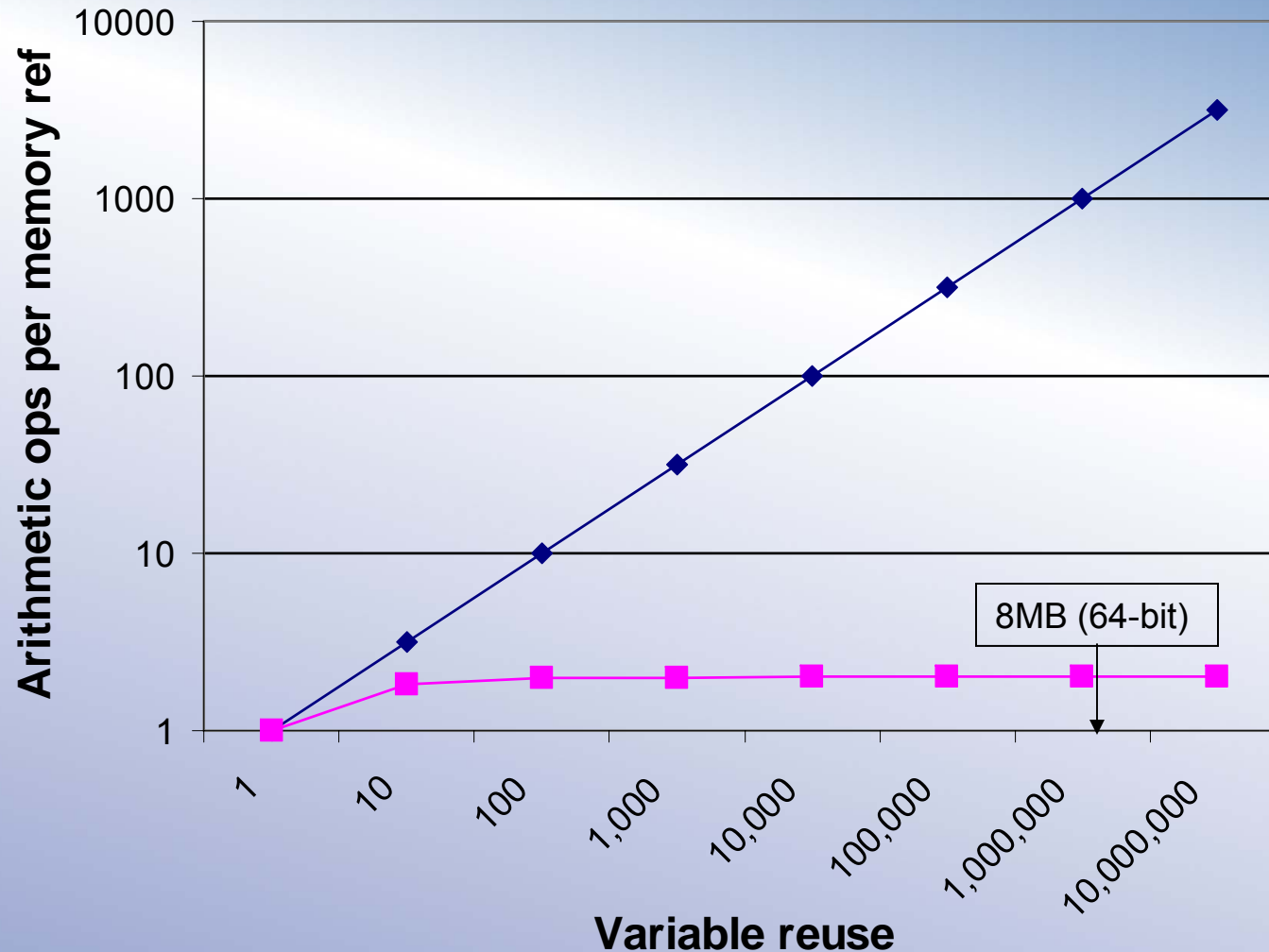
**CS@UH**

# Comparing Matrix-Matrix and Matrix-Vector Multiplication

- MV: Arithmetic  $\sim 2PQ$   
Memory  $\sim PQ(1+1/K)$
- MM: Arithmetic  $\sim 2PQR$   
Memory  $\sim PQR^2/\sqrt{K}$



# Impact of variable reuse



With an 8MB cache a memory bandwidth 1000 times smaller than the ALU bandwidth can be tolerated without starvation



COSC 6365  
Lecture 7  
2008-02-05

CS@UH

# Memory systems

- A nibble (4 bits) per DRAM chip
- 64-bit word requires 16 chips
- CAS Latency for DDR3 6 – 10 cycles, or for DDR3-1333 with 166 MHz/6ns clock 36 – 60 ns
- For a CAS Latency of 50 ns, the 16 chips can deliver 160 MB/sec
- A 2.5 GHz core with three 64-bit data paths can consume/produce  $3 \times 8 \times 2.5 \times 10^9 = 60$  GB/s

**And it get's worse for multi-core chips!**

Compare Matrix multiplication example

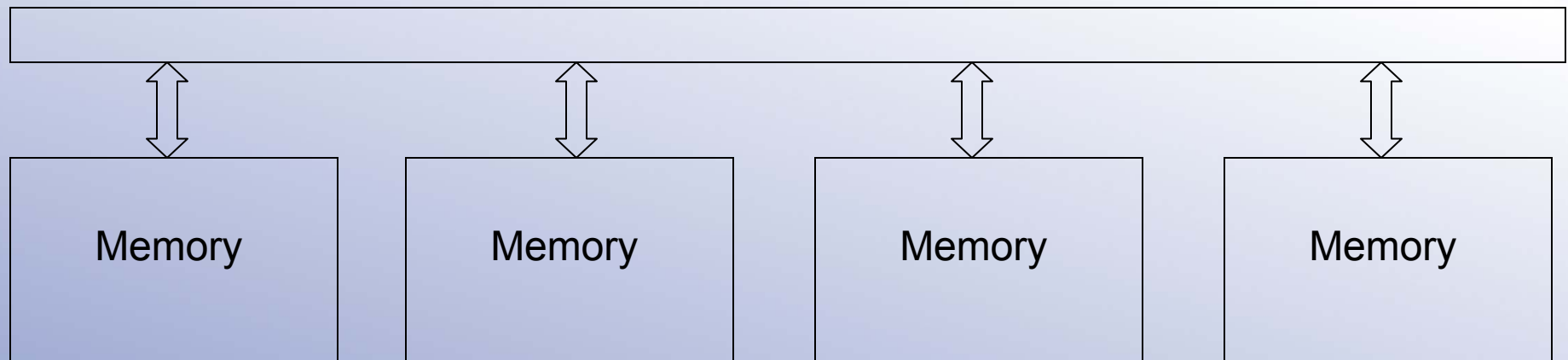


COSC 6365  
Lecture 7  
2008-02-05

**CS@UH**

# Memory Systems

- Interleaved memory
- Banked memory





COSC 6365  
Lecture 7  
2008-02-05

CS@UH

# Memory Systems

- **Synchronized access**
  - The banks are synchronized and accessed in parallel with the same local memory address for all banks. The retrieved data is latched for all banks and transmitted one word at a time.
- **Phased access**
  - In phased access, the memory banks operate independently. No synchronization between banks is required. After each bank has stored its result, a new access can begin. Different memory banks may use different local addresses.



# Memory Systems

- In a memory system the address is partitioned into a *bank number* and a *local address* within the bank.
- The typical way in which the bank number  $K$  and local address  $LA$  for an address  $A$  and  $B$  memory banks is determined is as follows:

$$K = A \bmod B$$

$$LA = \left\lfloor \frac{A}{B} \right\rfloor$$

- Successive indices are mapped to successive memory banks in a *cyclic* manner.
- Usually, the number of memory banks is a power of two. For  $2^k$  memory banks, the  $k$  lowest order bits determine the bank number, while the remaining higher order bits determine the local memory address.



# Memory Systems

## Synchronized Access

Example: 16 banks, cycle time = 10, access elements 31 – 65, element 0 in bank 0

Starting cycle	Bank number							
	0	1	2	3	.....	.....	14	15
10								31/10
20	32/20	33/21	34/22	35/23			46/34	47/35
36	48/36	49/37	50/38	51/39			62/50	63/51
52	64/52	65/53						

Total: 53 cycles to access elements 31 - 65



# Memory Systems

## Phased Access

Example: 16 banks, cycle time = 10, access elements 31 – 65,  
element 0 in bank 0

Starting cycle	Bank number							
	0	1	2	3	.....	.....	14	15
10	32/11	33/12	34/13	35/14			46/25	31/10
26	48/27	49/28	50/29	51/30			62/41	47/26
42	64/43	65/44						63/42

Total: 44 cycles to access elements 31 - 65



COSC 6365  
Lecture 7  
2008-02-05



# Memory Systems

Phased Access memory has better performance, sometimes much better performance

Example: Strided access with a stride of  $\#$  of banks-1

Synchronized Access:  $(\# \text{ of elements}) \times (\text{bank cycle time})$

Phased Access:  $(\# \text{ of elements}) + (\text{bank cycle time})$



COSC 6365  
Lecture 7  
2008-02-05

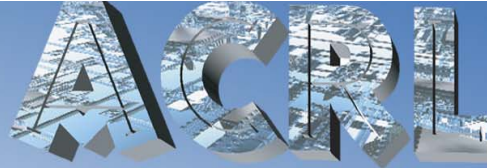
**CS@UH**

# Memory Systems

## Bank Conflicts

Stride is such that a bank is visited more frequently than bank cycle time

Worst case: Stride is a multiple of the number of banks



ADVANCED COMPUTING RESEARCH LABORATORY

COSC 6365  
Lecture 7  
2008-02-05



# Memory Systems

Bank Cycle time = 2

Arithmetic pipeline = 4

$$Y \leftarrow X + Z$$

Bank Number							
0	1	2	3	4	5	6	7
X(0)	X(1)	X(2)	X(3)	X(4)	X(5)	X(6)	X(7)
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	Z(0)	Z(1)	Z(2)	Z(3)	Z(4)	Z(5)
Z(6)	Z(7)	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	Y(0)	Y(1)	Y(2)	Y(3)
Y(4)	Y(5)	Y(6)	Y(7)	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--

Memory Accesses conflict free



# Memory Systems

Time	Bank Number								Pipeline stage			
	0	1	2	3	4	5	6	7	1	2	3	4
0	X(0)		Z(0)									
1	X(0)	X(1)	Z(0)	Z(1)								
2		X(1)	X(2)	Z(1)	Z(2)				0			
3			X(2)	X(3)	Z(2)	Z(3)			1	0		
4				X(3)	X(4)	Z(3)	Z(4)		2	1	0	
5					X(4)	X(5)	Z(4)	Z(5)	3	2	1	0
6	Z(6)				Y(0)	X(5)	X(6)	Z(5)	4	3	2	1
7	Z(6)				Y(0)	Y(1)	X(6)	X(7)	5	4	3	2
8		Z(7)				Y(1)	Y(2)	X(7)	6	5	4	3
9		Z(7)					Y(2)	Y(3)	7	6	5	4
10	Y(4)							Y(3)		7	6	5
11	Y(4)	Y(5)									7	6
12		Y(5)	Y(6)									7
13			Y(6)	Y(7)								
14				Y(7)								



# Memory Systems

Bank Cycle time = 2

Arithmetic pipeline = 4

$$Y \leftarrow X + Z$$

Bank Number							
0	1	2	3	4	5	6	7
X(0)	X(1)	X(2)	X(3)	X(4)	X(5)	X(6)	X(7)
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
Z(0)	Z(1)	Z(2)	Z(3)	Z(4)	Z(5)	Z(6)	Z(7)
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
Y(0)	Y(1)	Y(2)	Y(3)	Y(4)	Y(5)	Y(6)	Y(7)
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--

Memory Accesses conflicts



# Memory Systems

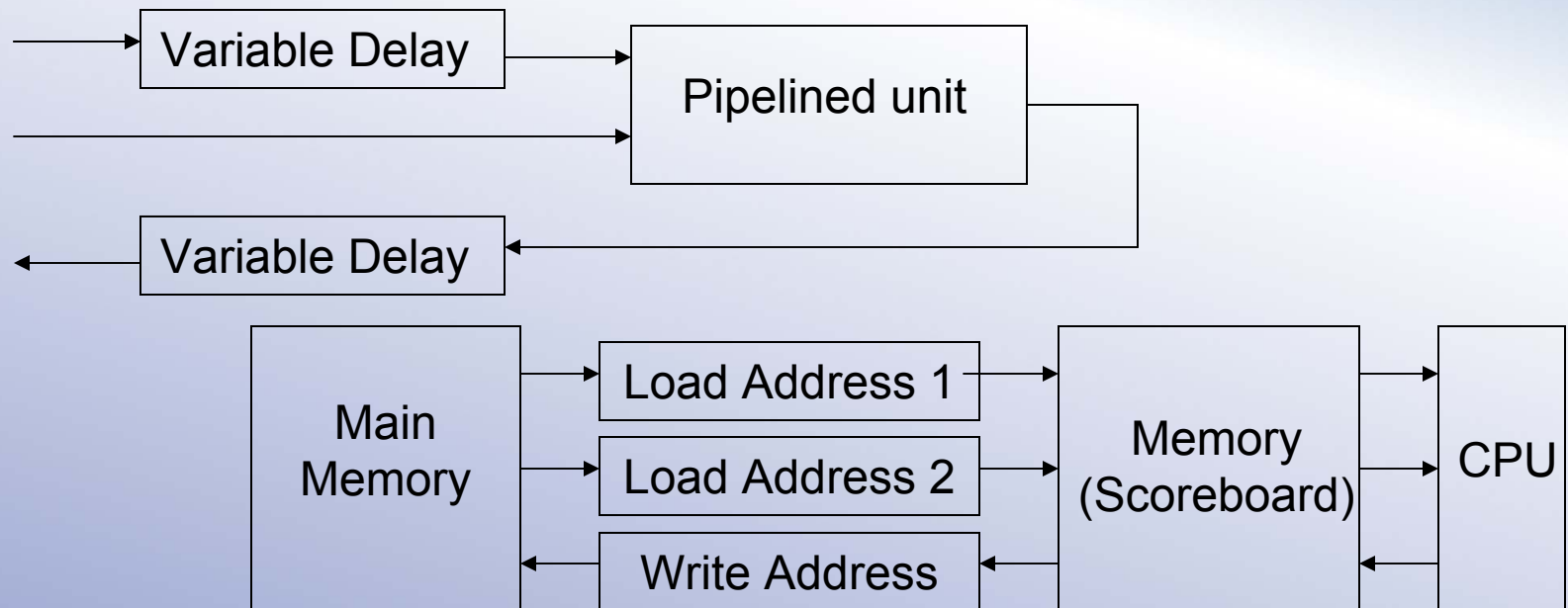
Time	Bank Number								Pipeline stage			
	0	1	2	3	4	5	6	7	1	2	3	4
0	X(0)											
1	X(0)	X(1)										
2	Z(0)	X(1)	X(2)									
3	Z(0)	Z(1)	X(2)	X(3)								
4		Z(1)	Z(2)	X(3)	X(4)				0			
5			Z(2)	Z(3)	X(4)	X(5)			1	0		
6				Z(3)	Z(4)	X(5)	X(6)		2	1	0	
7					Z(4)	Z(5)	X(6)	X(7)	3	2	1	0
8	Y(0)					Z(5)	Z(6)	X(7)	4	3	2	1
9	Y(0)	Y(1)					Z(6)	Z(7)	5	4	3	2
10		Y(1)	Y(2)					Z(7)	6	5	4	3
11			Y(2)	Y(3)					7	6	5	4
12				Y(3)	Y(4)					7	6	5
13					Y(4)	Y(5)					7	6
14						Y(5)	Y(6)					7
15							Y(6)	Y(7)				
16								Y(7)				



# Memory Systems

Memory conflicts may cause processor stalls!

Hardware (scoreboard) resolves alignment needs of operands and instructions





# Memory Systems

**Column Access  
Conflict free**  
**Row Access  
Conflicts**

Bank Number							
0	1	2	3	4	5	6	7
(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)
(8,0)	(9,0)	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	(79,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)	(7,1)
(8,1)	(9,1)	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	(79,1)
(0,2)	(1,2)	(3,2)	(4,2)	(5,2)	(6,2)	(6,2)	(7,2)
(8,2)	(9,2)	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	(79,2)
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--

80 x 80 array laid out in column major order



# Memory Systems

Row Access  
Conflict free

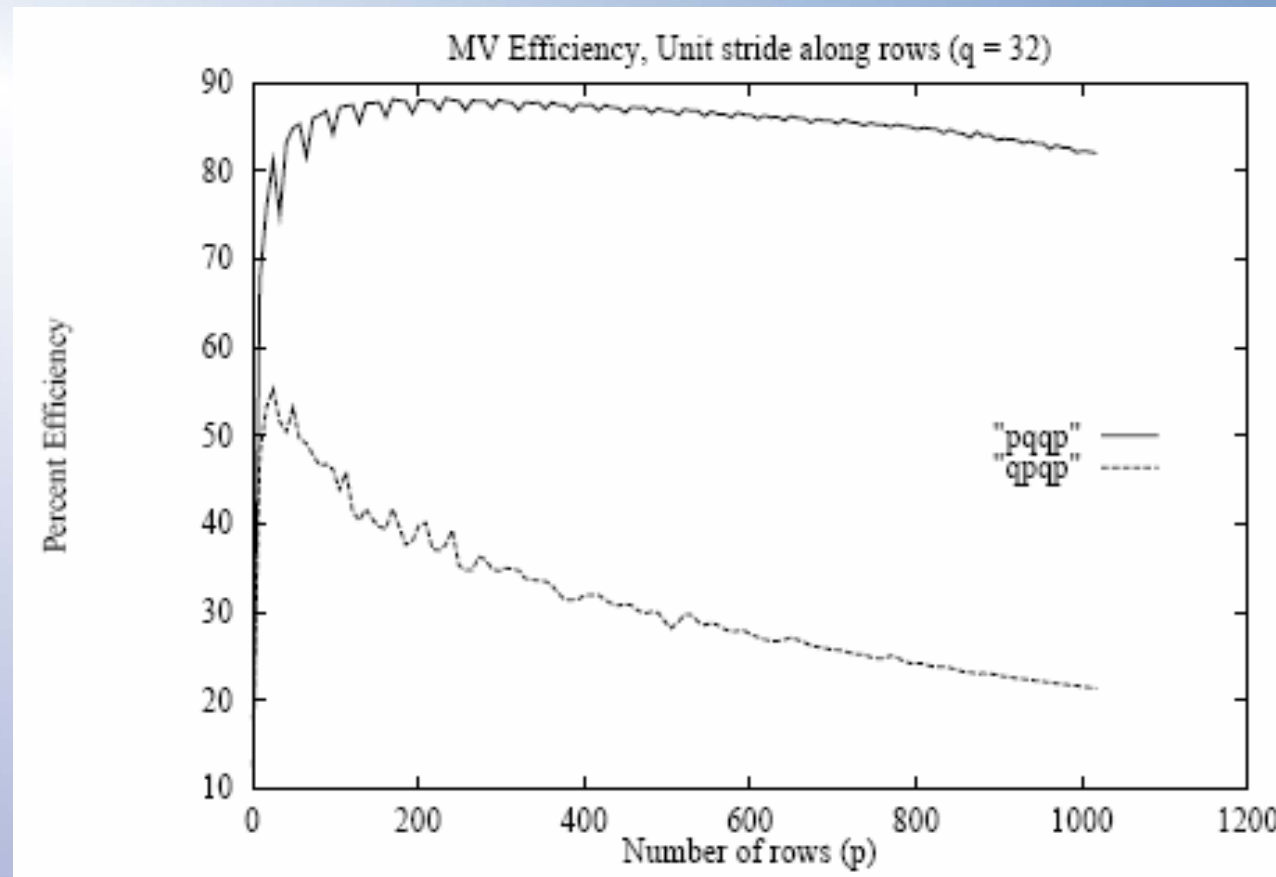
Column Access  
Conflicts

Bank Number							
0	1	2	3	4	5	6	7
(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
(0,8)	(0,9)	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	(0,79)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
(1,8)	(1,9)	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	(1,79)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
(2,8)	(2,9)	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	(2,79)
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--

80 x 80 array laid out in row major order



# Memory Systems



The impact of stride/layout on performance



# Memory Systems

## How to avoid bank conflicts?

Padding

Column Access  
Conflict free

Row Access  
Conflict free

Bank Number							
0	1	2	3	4	5	6	7
(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)
(8,0)	(9,0)	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	(79,0)
(80,1)	(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)
(7,1)	(8,1)	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	(78,1)
(79,1)	(80,1)	(0,2)	(1,2)	(2,2)	(3,2)	(4,2)	(5,2)
(6,2)	(7,2)	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	(77,2)
(78,2)	(79,2)	(80,2)	(0,3)	(1,3)	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--

Column major order



# Memory Systems

## How to avoid bank conflicts?

Padding

Row Access  
Conflict free

Column Access  
Conflict free

Bank Number							
0	1	2	3	4	5	6	7
(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
(0,8)	(0,9)	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	(0,79)
(0,80)	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)
(1,7)	(1,8)	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	(1,78)
(1,79)	(1,80)	(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)
(2,6)	(2,7)	--	--	--	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	(2,77)
(2,78)	(2,79)	(2,80)	(3,0)	(3,1)	--	--	--
--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--

Row major order



COSC 6365  
Lecture 7  
2008-02-05

**CS@UH**

# Memory Systems

- Padding in this case offers conflict free row and column access
  - by adding one row in column major order layout
  - by adding one column in row major order layout

Padding strategy depends on sequence to storage association

Padding wastes memory



COSC 6365  
Lecture 7  
2008-02-05

CS@UH

# Memory Systems

How to guarantee conflict free access?

The largest number of elements  $L$  that can be accessed with a stride of  $S$  without conflict from  $B$  banks is determined by

$$L \gcd(S, B) \leq B$$

What about conflict free access to rows, columns, diagonals, anti-diagonals, and blocks?



# Memory systems

Example:  $(\delta_1, \delta_2)$ -skewing for two dimensional arrays  
in which element  $(i,j)$  is mapped to  $\delta_1 i + \delta_2 j$

Row Access:  $L \gcd(\delta_1, B) \leq B$

Column Access:  $L \gcd(\delta_2, B) \leq B$

Diagonal Access:  $L \gcd(\delta_1 + \delta_2, B) \leq B$

Anti-diagonal Access:  $L \gcd(\delta_1 - \delta_2, B) \leq B$

With proper skewing square blocks of size  $L$   
can be access conflict free



COSC 6365  
Lecture 7  
2008-02-05



# Memory Systems

For  $B$  even not all these four access modes can support  $L = B$ ; the best is  $L = B/2$

For  $\delta_1$  and  $B$  both even  $L \leq B/2$  for row access

For  $\delta_1$  odd and  $B$  even  $L \leq B$  for row access

For  $\delta_2$  and  $B$  both even  $L \leq B/2$  for column access

For  $\delta_2$  odd and  $B$  even  $L \leq B$  for column access

If both  $\delta_1$  and  $\delta_2$  are odd then both  $\delta_1 + \delta_2$  and  $\delta_1 - \delta_2$  are even and for  $B$  even  $L \leq B/2$  for both diagonal and anti-diagonal access



# (3,2)-skewing for row major ordering

8 x 8 matrix

Row access:  $L=B/2$

Column access:  $L=B$

Diagonal access:  $L=B$

Anti-diag. access:  $L=B$

Memory utilization 50%,  
but can be improved  
fairly easily, e.g. by  
packing pairs of rows

		Bank Number							
		0	1	2	3	4	5	6	7
(0,0)	--	(0,1)	--	(0,2)	--	(0,3)	--	(0,4)	--
(0,4)	--	(0,5)	--	(0,6)	--	(0,7)	--	(0,8)	--
--	--	--	(1,0)	--	(1,1)	--	(1,2)	--	(1,3)
--	(1,3)	--	(1,4)	--	(1,5)	--	(1,6)	--	(1,7)
--	(1,7)	--	--	--	--	--	(2,0)	--	(2,1)
(2,1)	--	(2,2)	--	(2,3)	--	(2,4)	--	(2,5)	--
(2,5)	--	(2,6)	--	(2,7)	--	--	--	--	--
--	(3,0)	--	(3,1)	--	(3,2)	--	(3,3)	--	(3,4)
--	(3,4)	--	(3,5)	--	(3,6)	--	(3,7)	--	(3,8)
--	--	--	--	(4,0)	--	(4,1)	--	(4,2)	--
(4,2)	--	(4,3)	--	(4,4)	--	(4,5)	--	(4,6)	--
(4,6)	--	(4,7)	--	--	--	--	--	--	(5,0)
--	(5,1)	--	(5,2)	--	(5,3)	--	(5,4)	--	(5,5)
--	(5,5)	--	(5,6)	--	(5,7)	--	(5,8)	--	(5,9)
--	--	(6,0)	--	(6,1)	--	(6,2)	--	(6,3)	--
(6,3)	--	(6,4)	--	(6,5)	--	(6,6)	--	(6,7)	--
(6,7)	--	--	--	--	--	(7,0)	--	(7,1)	--
--	(7,2)	--	(7,3)	--	(7,4)	--	(7,5)	--	(7,6)
--	(7,6)	--	(7,7)	--	--	--	--	--	--



# Skewing

Skewing introduces alignment issues

Bank	0	1	2	3	4	5	6	7
Diag	(0,0)	(5,5)	(2,2)	(7,7)	(4,4)	(1,1)	(6,6)	(3,3)

Diagonal elements appear out of order (stride  $\delta_1 + \delta_2$ )

Bank	0	1	2	3	4	5	6	7
Diag	(0,0)	(3,0)	(6,0)	(1,0)	(4,0)	(7,0)	(2,0)	(5,0)

Column elements appear out of order (stride  $\delta_1$ )

Bank	0	1	2	3	4	5	6	7
Diag	(4,2)	(7,2)	(2,2)	(5,2)	(0,2)	(3,2)	(6,2)	(1,2)

Column elements appear out of order (with offset and stride  $\delta_1$ )



COSC 6365  
Lecture 7  
2008-02-05



# Memory Systems

For  $B$  even not all these four access modes can support  $L = B$ ; the best is  $L = B/2$

How to improve it?

**Make  $B$  odd!**



# Memory Systems

$(2^l, 1)$ -skewing for  $B=2^{2l}+1$   
allows conflict free access to  
rows, columns, diagonals,  
anti-diagonals, and square  
blocks of size  $2^{2l}$

$$K = (2^l + j) \bmod B$$

$$LA = i \left\lfloor \frac{N}{B} \right\rfloor + \left\lfloor \frac{j}{B} \right\rfloor$$

Bank number				
0	1	2	3	4
(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(0,5)	(0,6)	(0,7)	--	--
(1,3)	(1,4)	(1,0)	(1,1)	(1,2)
--	--	(1,5)	(1,6)	(1,7)
(2,1)	(2,2)	(2,3)	(2,4)	(2,0)
(2,6)	(2,7)	--	--	(2,5)
(3,4)	(3,0)	(3,1)	(3,2)	(3,3)
--	(3,5)	(3,6)	(3,7)	--
(4,2)	(4,3)	(4,4)	(4,0)	(4,1)
(4,7)	--	--	(4,5)	(4,6)
(5,0)	(5,1)	(5,2)	(5,3)	(5,4)
(5,5)	(5,6)	(5,7)	--	--
(6,3)	(6,4)	(6,0)	(6,1)	(6,2)
--	--	(6,5)	(6,6)	(6,7)
(7,1)	(7,2)	(7,3)	(7,4)	(7,0)
(7,6)	(7,7)	--	--	(7,5)



# Memory Systems

$(2^l, 1)$ -skewing for  $B=2^{2^l+1}-1$  also allows conflict free access to rows, columns, diagonals, anti-diagonals, and square blocks of size  $2^{2^l}$

Choosing  $B=2^{2^l}-1$  instead of  $B=2^{2^l}+1$ , or  $B=2^{2^l+1}+1$  instead of  $B=2^{2^l+1}-1$  for  $(2^l, 1)$ -skewing reduces  $L$ , the number of elements that can accessed conflict free.

**Guarantee for conflict free access:**

**Make  $B$  = prime number!**

**The Burroughs Scientific Processor!**



# Memory Systems

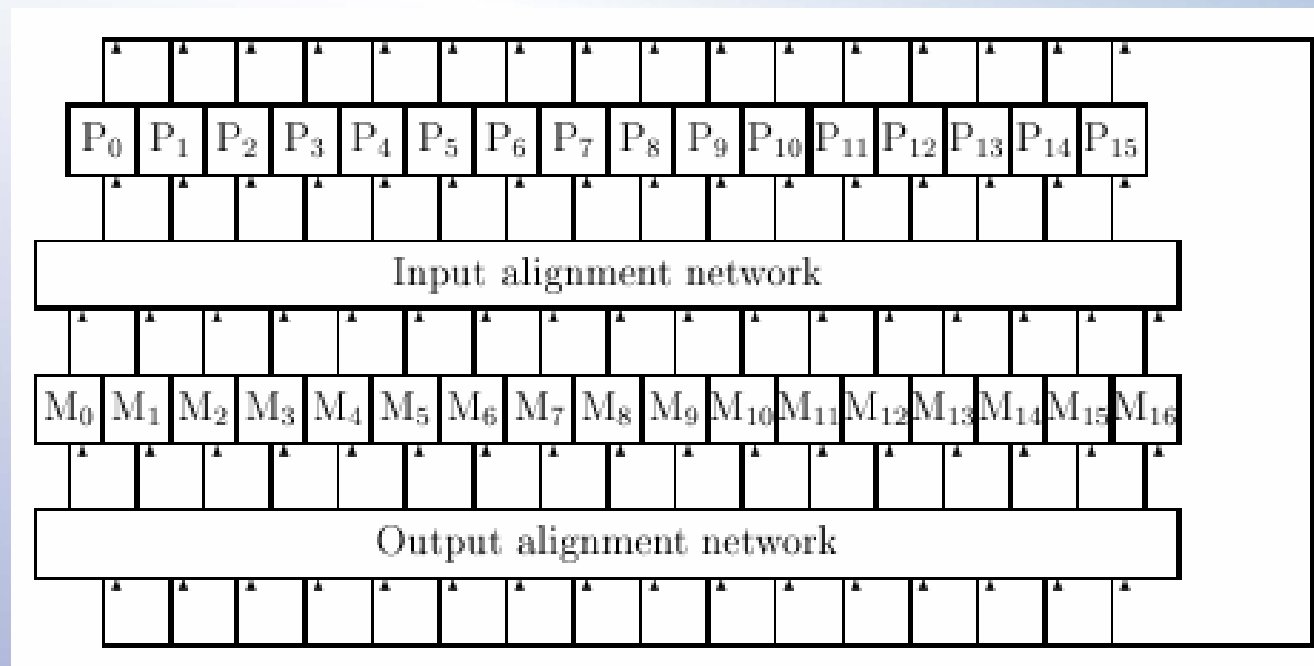
$\ell$	$2^{2\ell}+1$	Prime?	$2^{2\ell+1}-1$	Prime?
1	5	√	7	√
2	17	√	31	√
3	65		127	√
4	257	√	511	
5	1025		2047	
6	4097		8191	√
7	16385		32767	

The Burroughs Scientific Processor was designed for B=17 and L=16 for 16 processors



# Memory Systems

## The Burroughs Scientific Processor





COSC 6365  
Lecture 7  
2008-02-05

CS@UH

# Memory Systems

How to reduce conflicts for memory access for  $B = 2^k$  for some  $k$ ?

Randomize the sequence to storage association!

The Tera computer



<http://www.ai.mit.edu/projects/aries/course/notes/tera.pdf>

[http://inst.eecs.berkeley.edu/~n252/su06/Lorenzo\\_Midterms\\_MTA.pdf](http://inst.eecs.berkeley.edu/~n252/su06/Lorenzo_Midterms_MTA.pdf)