EE 357 Unit 13

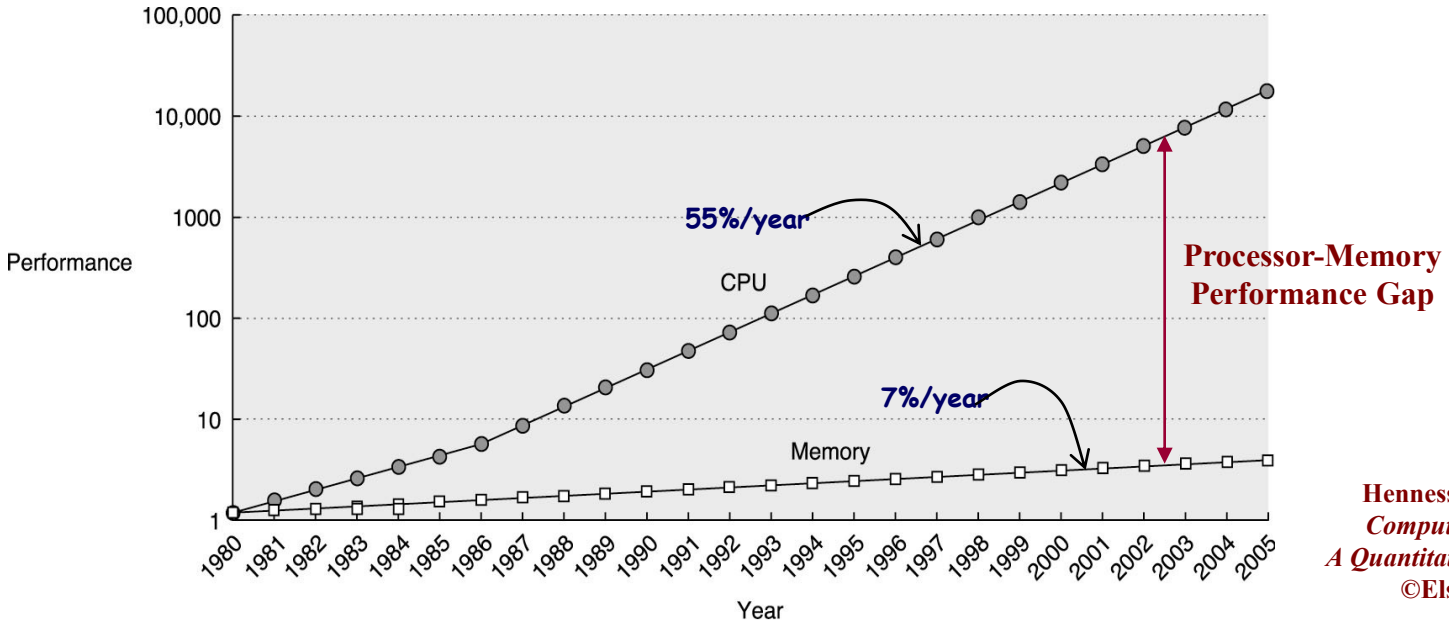
Memory System Overview

SRAM vs. DRAM

DMA & Endian-ness

The Memory Wall

- Problem: The Memory Wall
  - Processor speeds have been increasing much faster than memory access speeds (Memory technology targets density rather than speed)
  - Large memories yield large address decode and access times
  - Main memory is physically located on separate chips and inter-chip signals have much higher propagation delay than intra-chip signals



Hennessy and Patterson,
Computer Architecture –
A Quantitative Approach (2003)
©Elsevier Science

Improving Memory Performance

- Possibilities for improvement
  - Technology
    - Can we improve our transistor-level design to create faster RAMs
    - Can we integrate memories on the same chip as our processing logic
  - Architectural
    - Can we organize memory in a more efficient manner (this is our focus)

DRAM & SRAM

Memory Technology

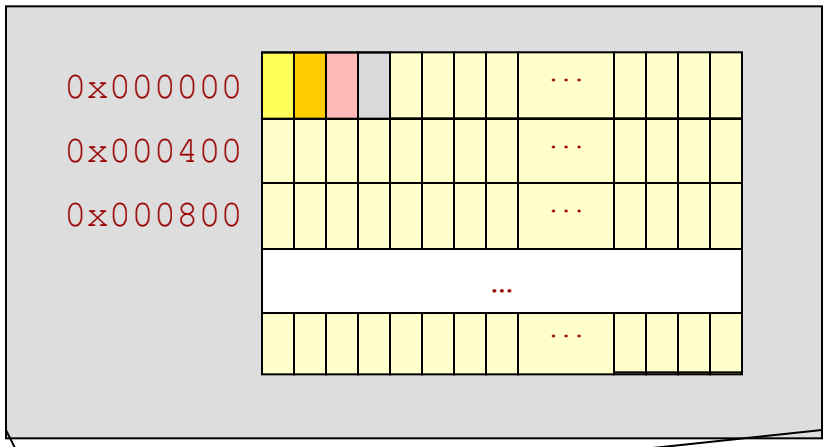
- Static RAM (SRAM)
  - Will retain values indefinitely (as long as power is on)
  - Stored bit is actively "remembered" (driven and regenerated by circuitry)
- Dynamic RAM (DRAM)
  - Will lose values if not refreshed periodically
  - Stored bit is passively "remembered" and needs to be regenerated by external circuitry

Memory Array

- Logical View = 1D array of rows (words)
  - Already this is 2D because each word is 32-bits (i.e. 32 columns)
- Physical View = 2D array of rows and columns
  - Each row may contain 1000's of columns (bits) though we have to access at least 8- (and often 16-, 32-, or 64-) bits at a time

0x0000	123489AB
0x0004	AB4982FE
0x0008	89AB97CD
0x000c	C8004DB2

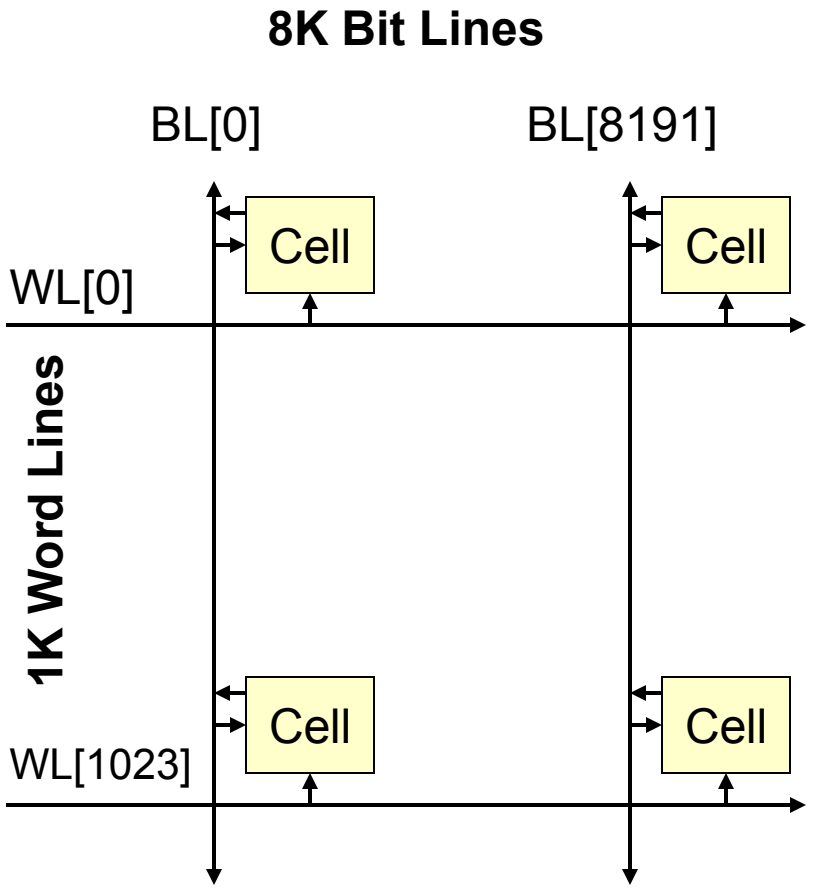
1D Logical View (Each row is a single word = 32-bits)



2D Physical View (e.g. a row is 1KB = 8Kb)

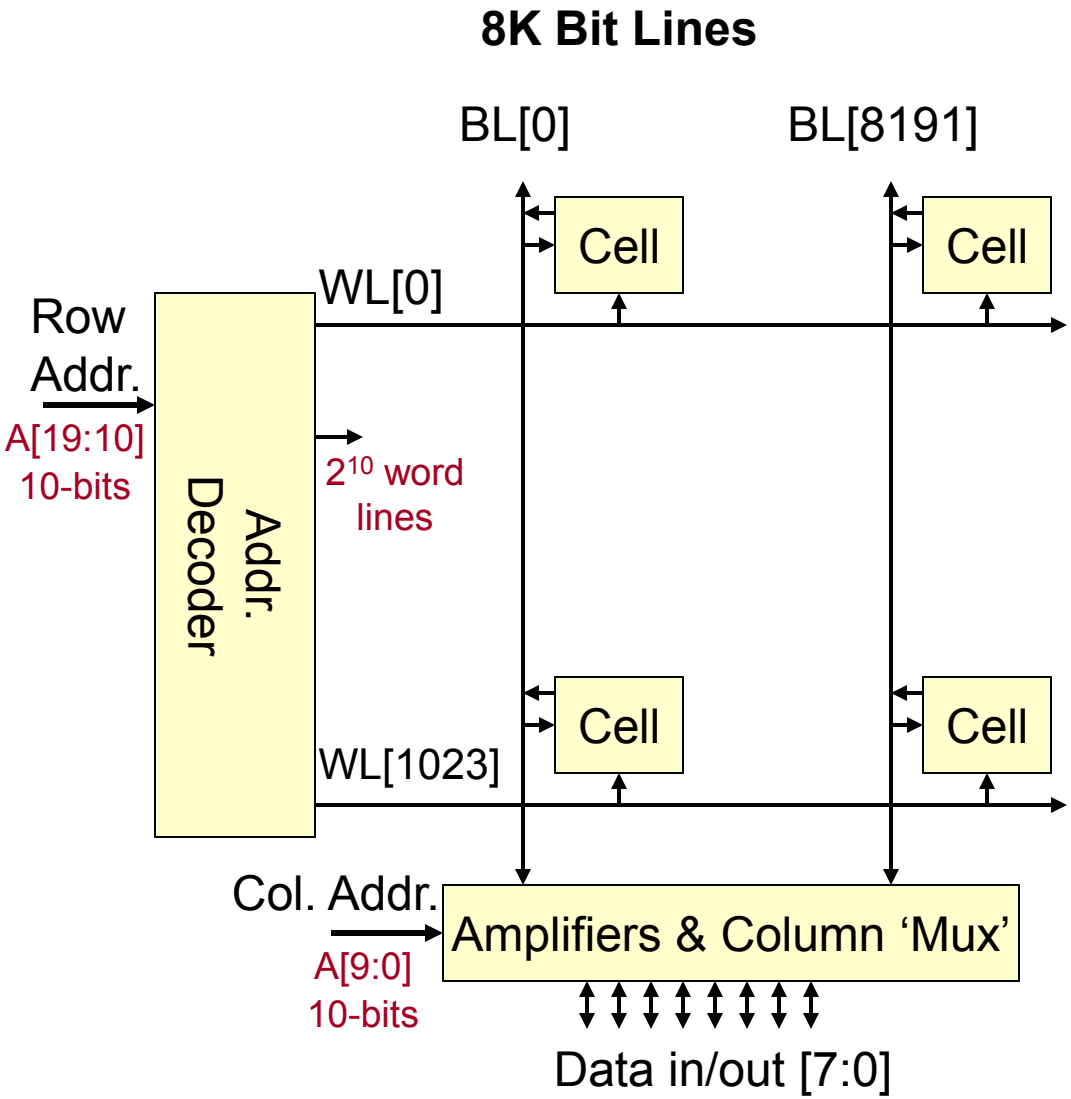
1Mx8 Memory Array Layout

- Start with array of cells that can each store 1-bit
- 1 MB = 8 Mbit = 2^{23} total cells
- This can be broken into a 2D array of cells (2^{10} x 2^{13})
- Each row connects to a WL = word line which selects that row
- Each column connects to a BL = bit line for read/write data



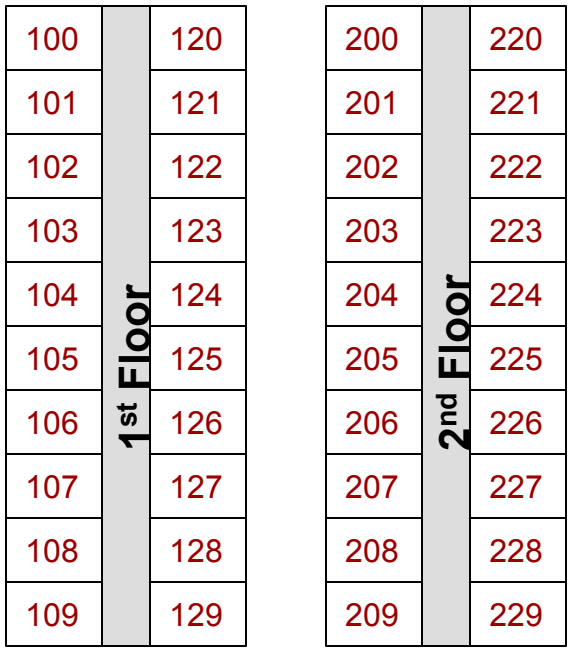
Row and Column Address

- For 1MB we need 20 address bits
- 10 upper address bits can select one of the 1024 rows
- Suppose we always want to read/write an 8-bits (byte), then we will group each set of 8-bits into 1024 byte columns (1024*8=8K)
- 10 lower address bits will select the column



Multidimensional Indexing from a Linear Address

- Analogy: Hotel room layout/addressing
  - Hotel Rooms 100 – 229 (linear range / 1-dimensional sequence)
  - Individual (or groups of) digits used for multi-dimensional indexing
    - First digit = Floor (Z dimension)
    - Second digit = Aisle/Column (X dimension)
    - Third digit = Row (Y dimension)



Analogy: Hotel Rooms

1st Digit = Floor
2nd Digit = Aisle
3rd Digit = Room w/in aisle

To refer to the range of rooms on the second floor, left aisle we would just say rooms **20x**

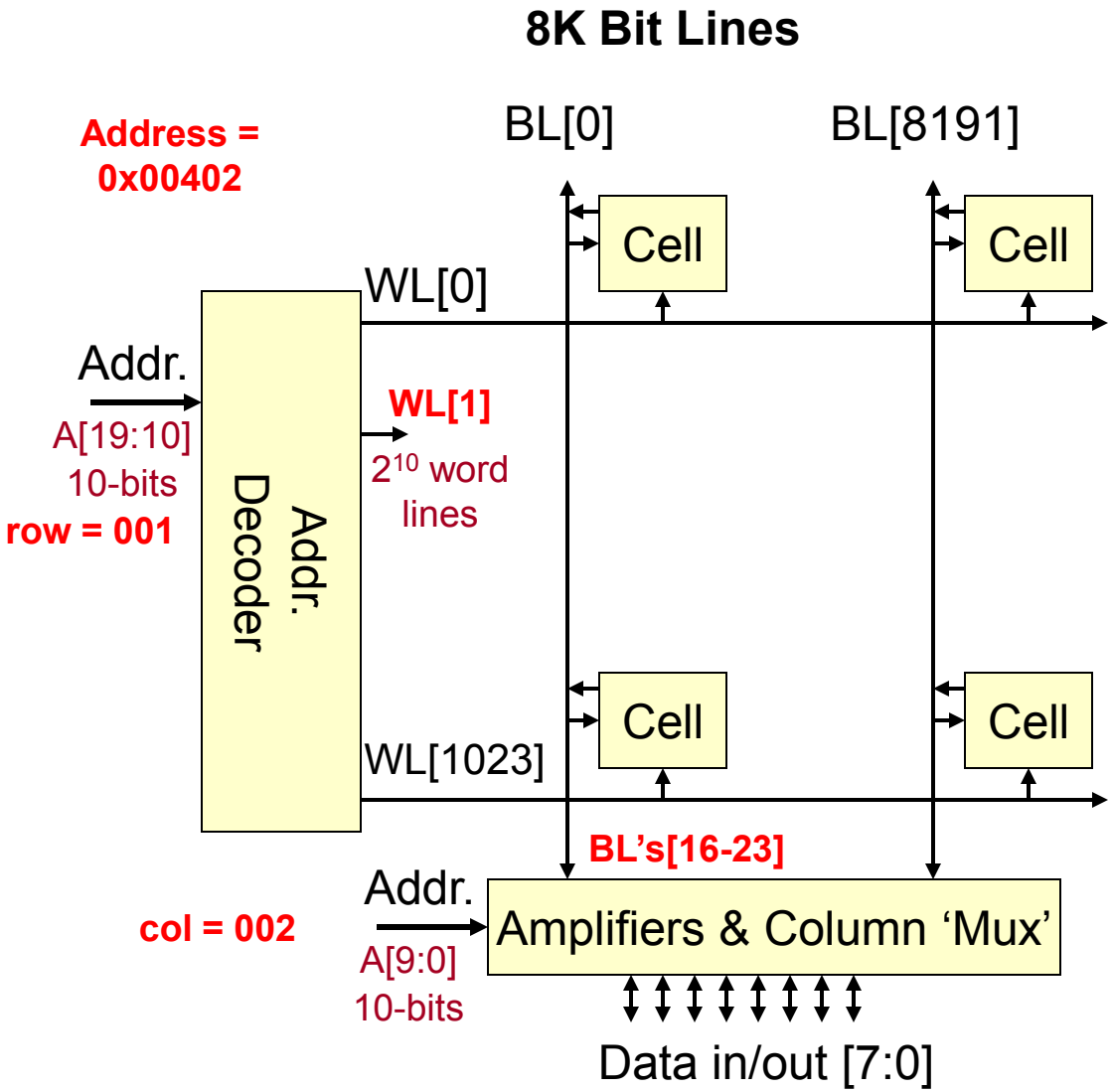
20 address bits =>
10 for row / 10 for column

Addr.	Row	Col
0x00402	0000 0000 01	00 0000 0010

R=1, C=2

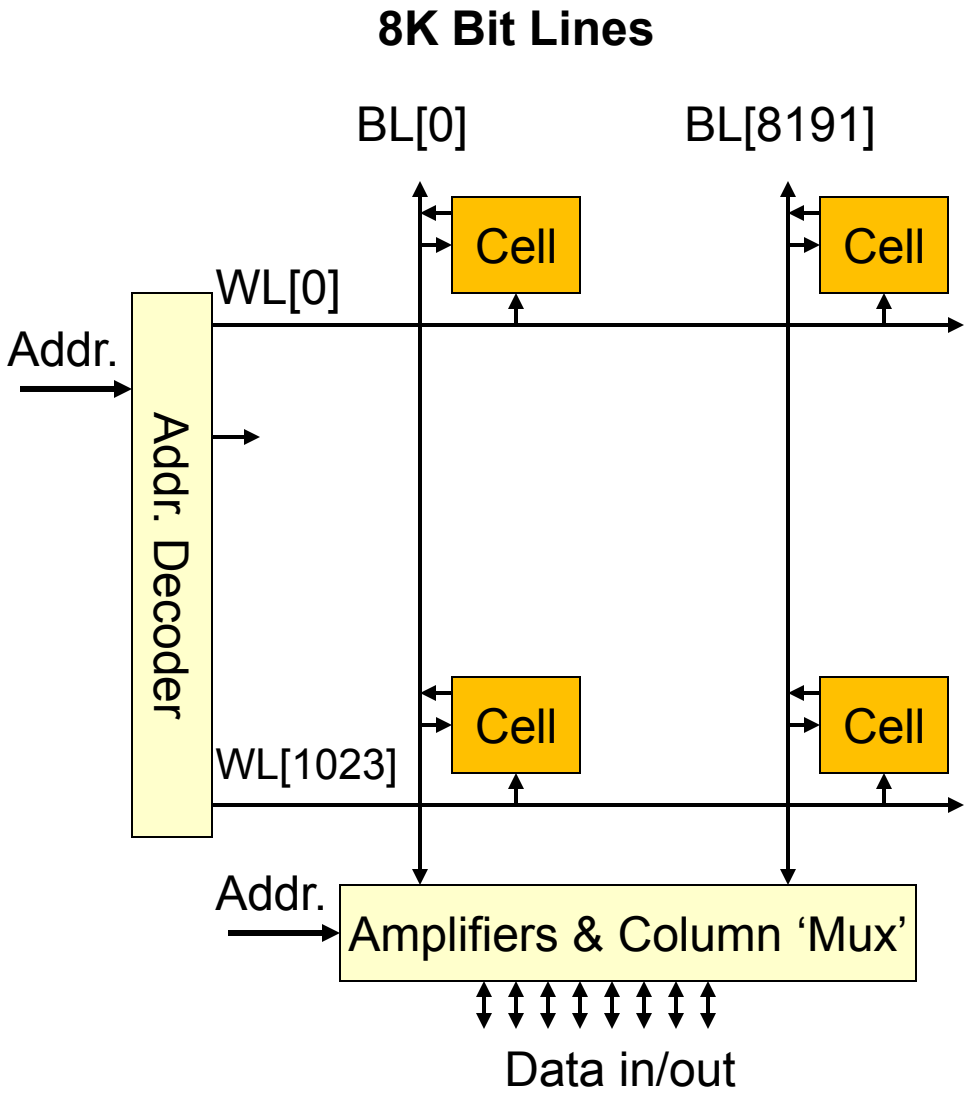
Periphery Logic

- Address decoders selects one row based on the input address number
- Column multiplexers use the address bits to select the right set of 8-bits from the 8K bit lines
- Bit lines are bidirectional lines for reading (output) or writing (input)



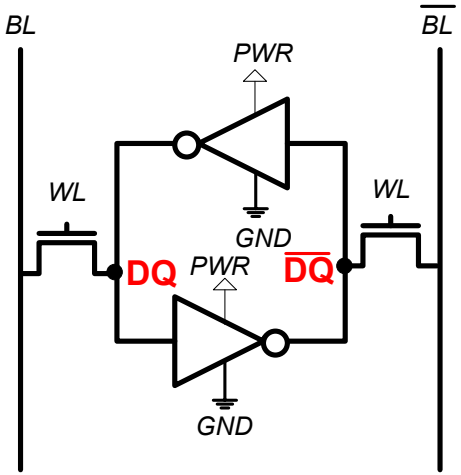
Memory Technologies

- Memory technologies share the same layout but differ in their cell implementation
- Static RAM (SRAM)
  - Will retain values indefinitely (as long as power is on)
  - When read, the stored bit is actively driven onto the bit lines
- Dynamic RAM (DRAM)
  - Will lose values if not refreshed periodically
  - When read, the stored bit passively pulls the bit line voltage up or down slightly and needs to be amplified

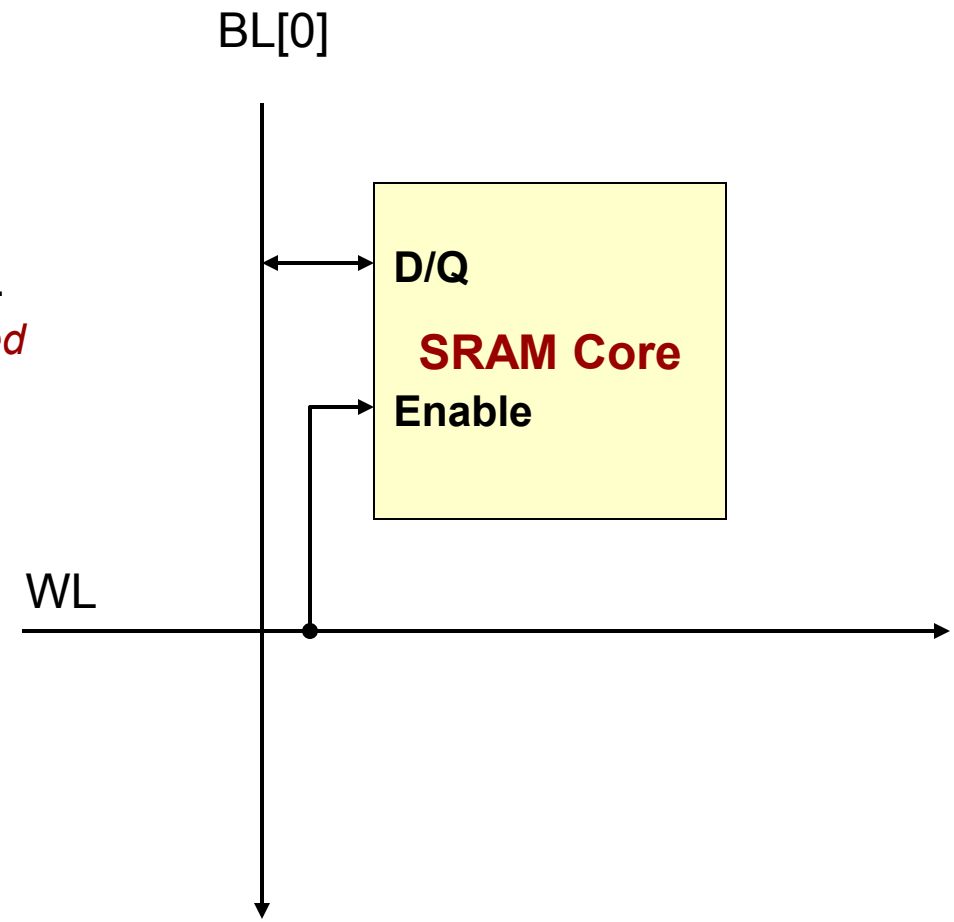


SRAM Cell

- Each memory cell requires **6 transistors**
- Each cell consists of a D-Latch is made from cross connected inverters which have active connections to PWR and GND.
  - Thus, the signal is *remembered* and *regenerated* as long as power is supplied

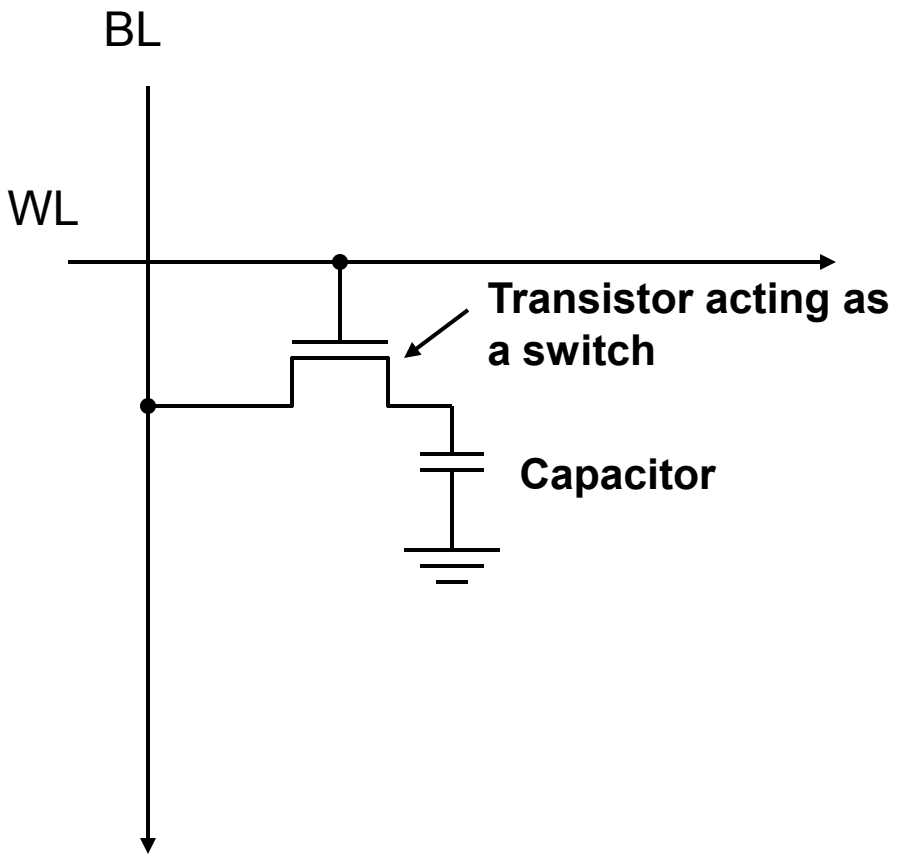


SRAM core implementation



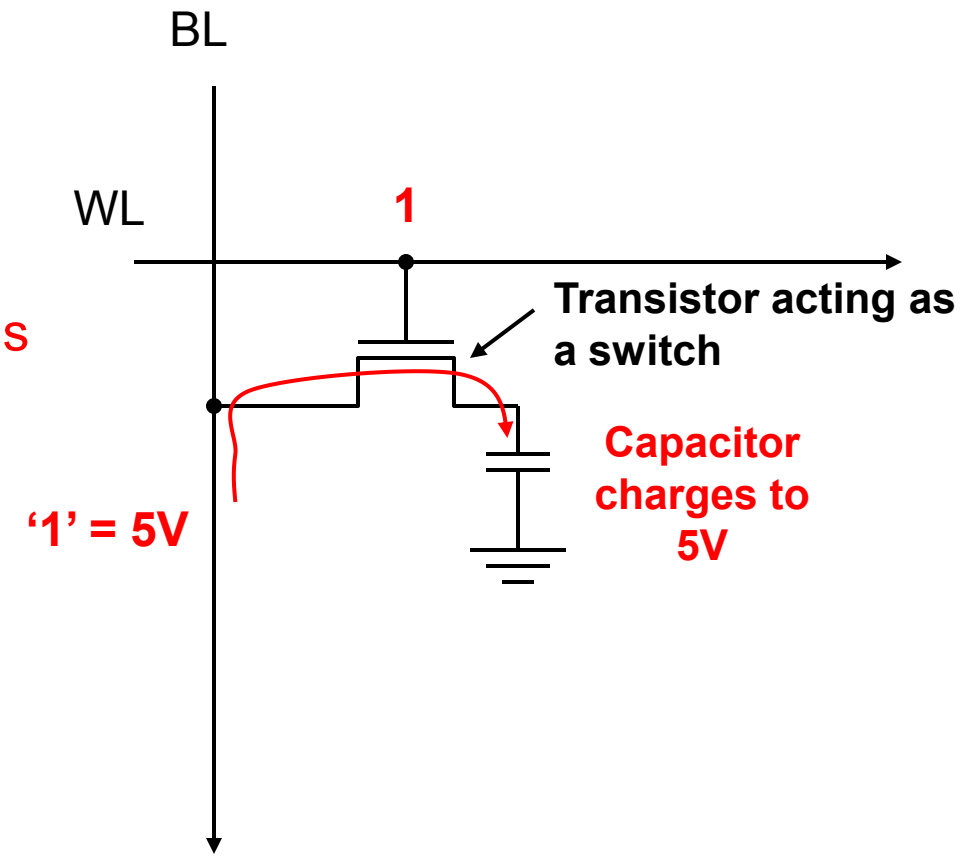
DRAM Cell

- Bit is stored on a capacitor and requires only **1 transistor and a capacitor**



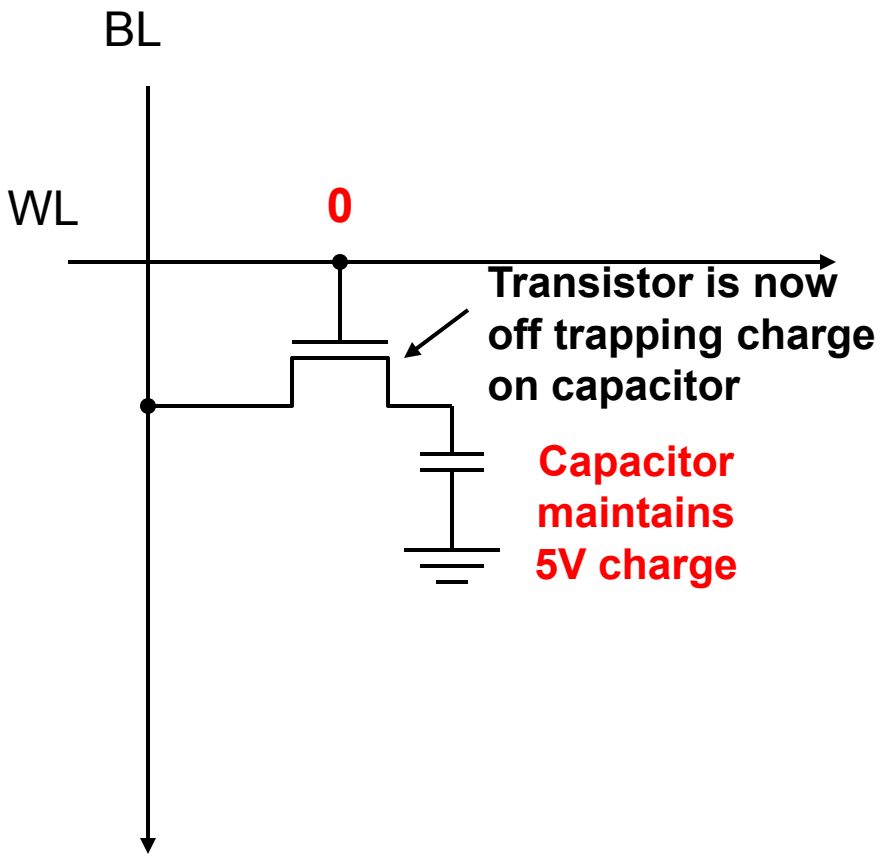
DRAM Cell

- Bit is stored on a capacitor and requires only **1 transistor**
- Write
  - WL=1 connects the capacitor to the BL which is driven to 1 or 0 and charges/discharges capacitor based on the BL value



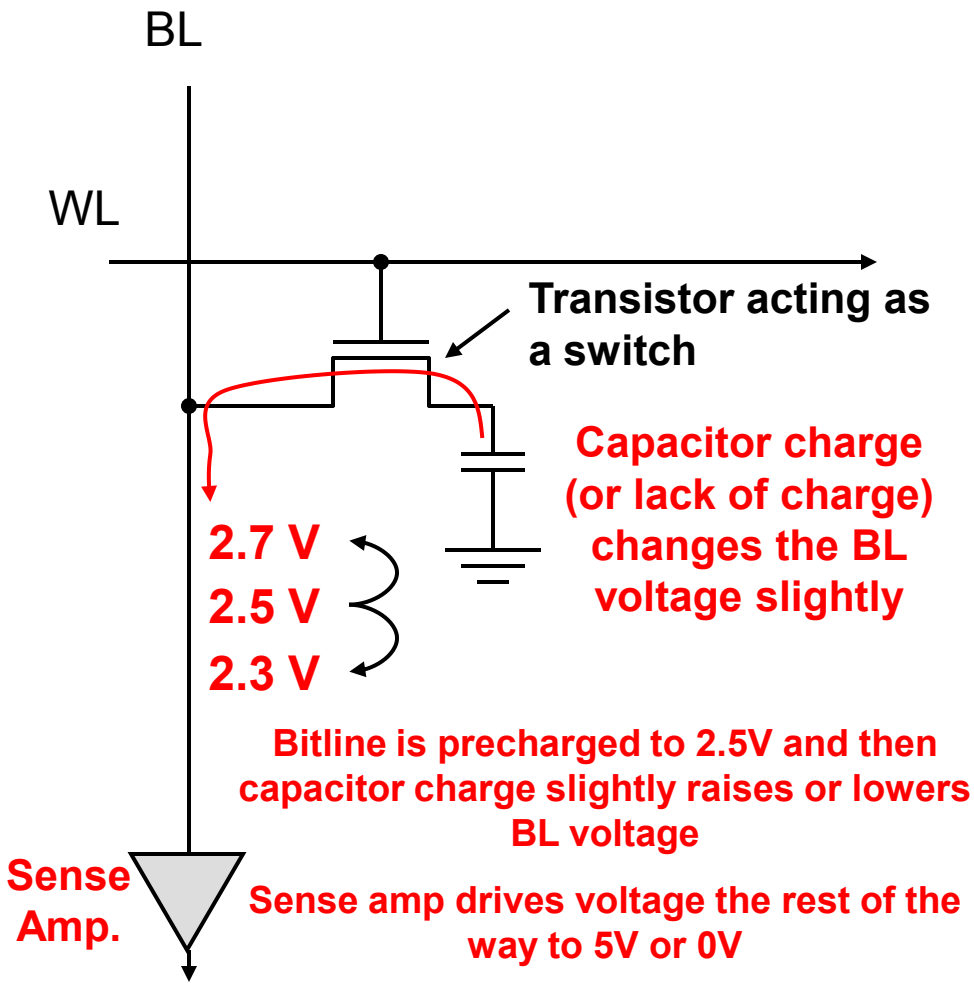
DRAM Cell

- Bit is stored on a capacitor and requires only **1 transistor**
- Write
  - WL=1 connects the capacitor to the BL which is driven to 1 or 0 and charges/discharges capacitor based on the BL value
- With WL=0 transistor is closed and value stored on the capacitor



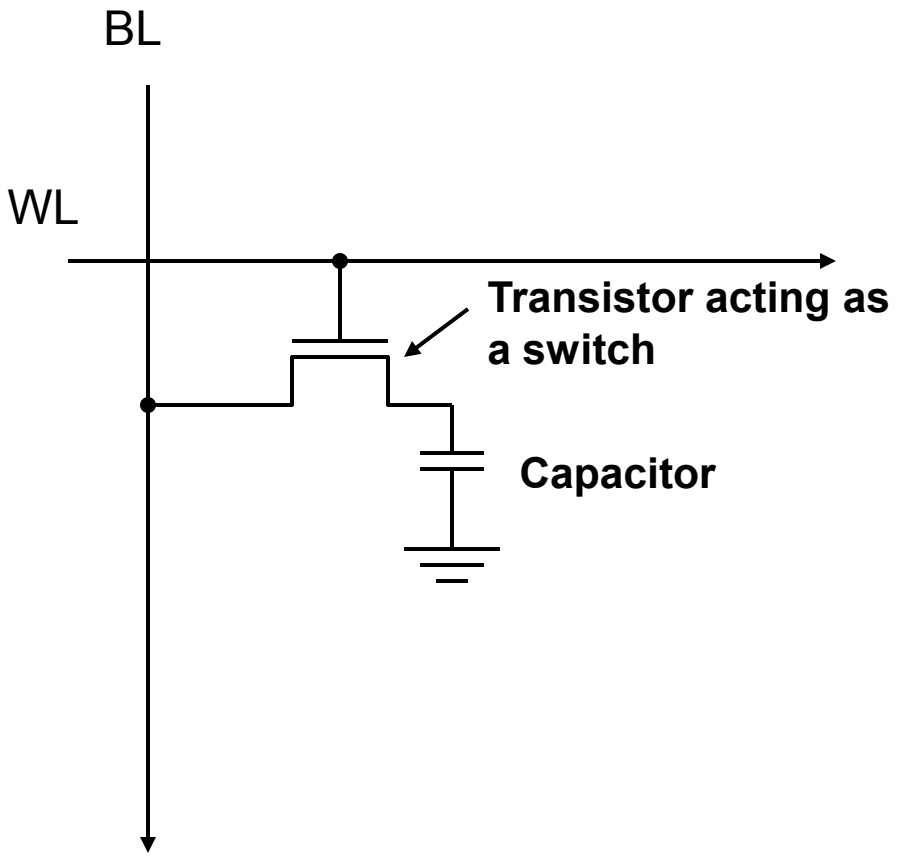
DRAM Cell

- Bit is stored on a capacitor and requires only **1 transistor**
- Write
  - WL=1 connects the capacitor to the BL which is driven to 1 or 0 and charges/discharges capacitor based on the BL value
- Read
  - BL is precharged to 2.5 V
  - WL=1 connects the capacitor to the BL allowing charge on capacitor to change the voltage on the BL



DRAM Issues

- Destructive Read
  - Charge is lost when capacitor value is read
  - Need to write whatever is read back to the cap.
- Leakage Current
  - Charge slowly leaks off the cap.
  - Value must be refreshed periodically



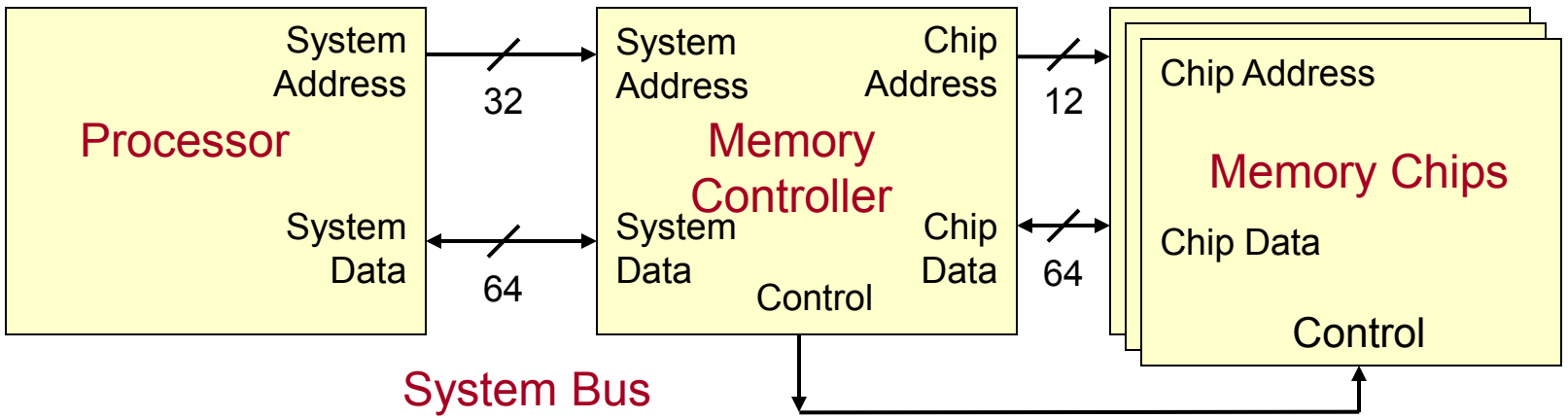
SRAM vs. DRAM Summary

- SRAM
  - Faster because bit lines are actively driven by the D-Latch
  - Faster, simpler interface due to lack of refresh
  - Larger Area for each cell which means less memory per chip
  - Used for cache memory where speed/latency is key
- DRAM
  - Slower because passive value (charge on cap.) drives BL
  - Slower due to refresh cycles
  - Small area means much greater density of cells and thus large memories
  - Used for main memory where density is key

MAIN MEMORY ORGANIZATION

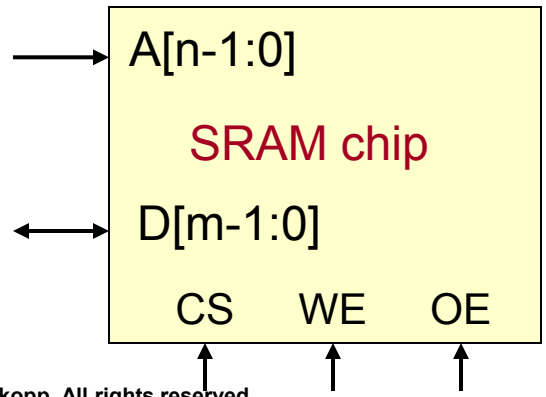
Architectural Block Diagram

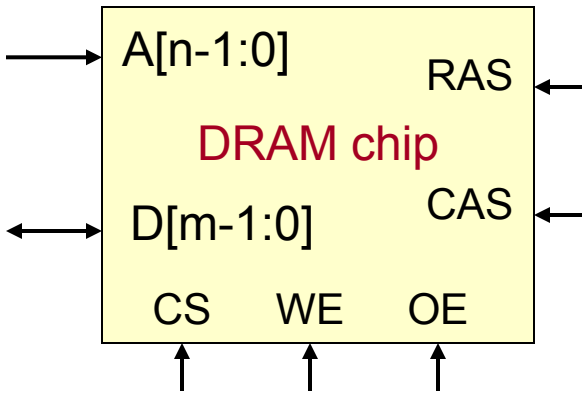
- Memory controller interprets system bus transactions into appropriate chip address and control signals
  - System address and data bus sizes do not have to match chip address and data sizes



Memory Block Diagrams

- SRAM
  - Bidirectional, shared data bus (only 1 chip can drive bus at a time)
  - CS = Chip select (enables the chip…in the likely case that multiple chips connected to bus)
  - WE / OE = Write / Output Enable (write the data or read data to/from the given address)
- DRAM
  - Bidirectional, shared data bus (only 1 chip can drive bus at a time)
  - RAS / CAS = Row / Column address strobe. Address broken into two groups of bits for the row and column in the 2D memory array (This allows address bus to be approx. half as wide as an SRAM's)
  - CS/WE/OE = Chip select & Write / Output Enable



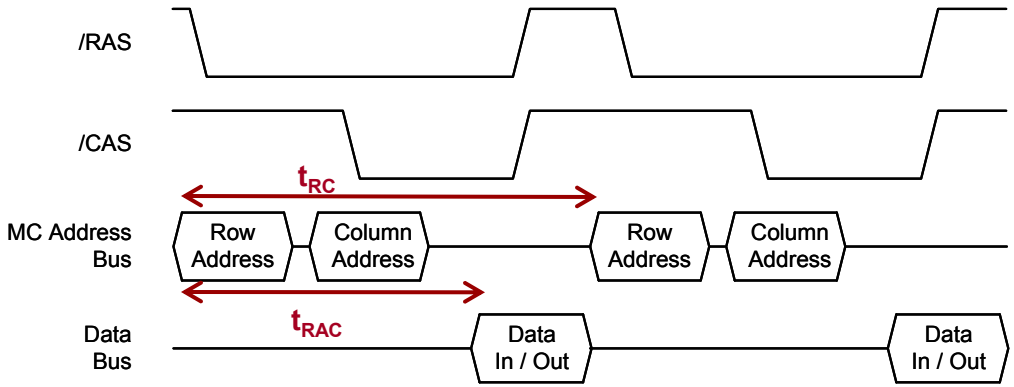


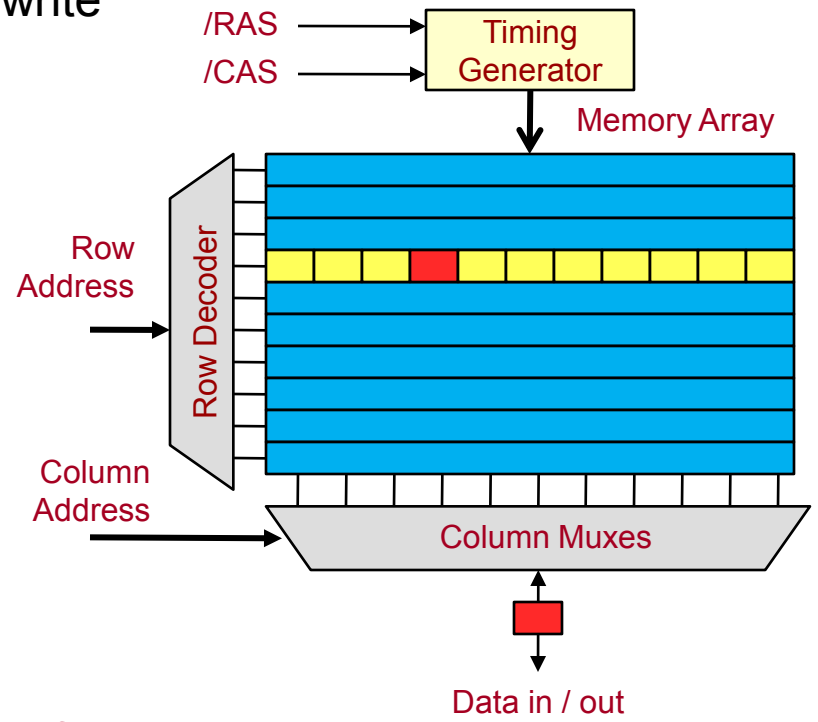
Implications of Memory Technology

- Memory latency of a single access using current DRAM technology will be slow
- We must improve bandwidth
  - Idea 1: Access more than just a single word at a time
  - Technology: EDO, SDRAM, etc.
  - Idea 2: Increase number of accesses serviced in parallel (in-flight accesses)
  - Technology: Banking

Legacy DRAM Timing

- Can have only a single access "in-flight" at once
- Memory controller must send row and column address portions for each access
  - RAS active holds the row open for reading/writing
  - CAS active selects the column to read/write





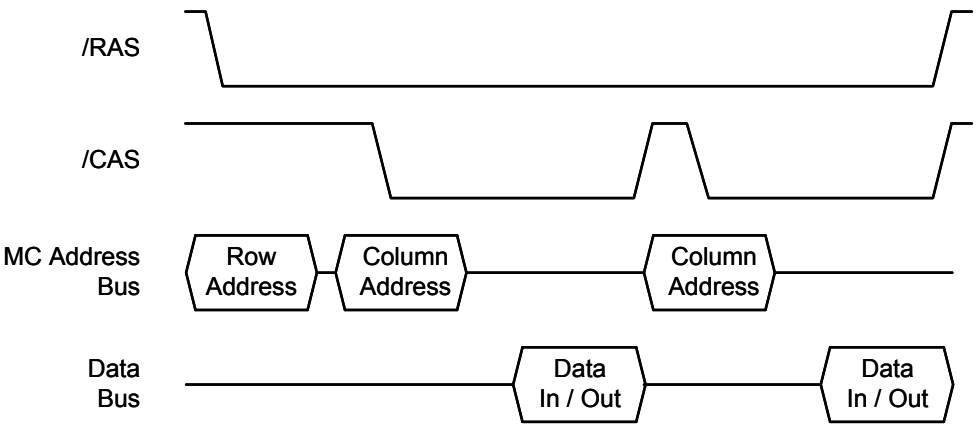
t_{RC}= Cycle Time (110ns) = Time before next access can start

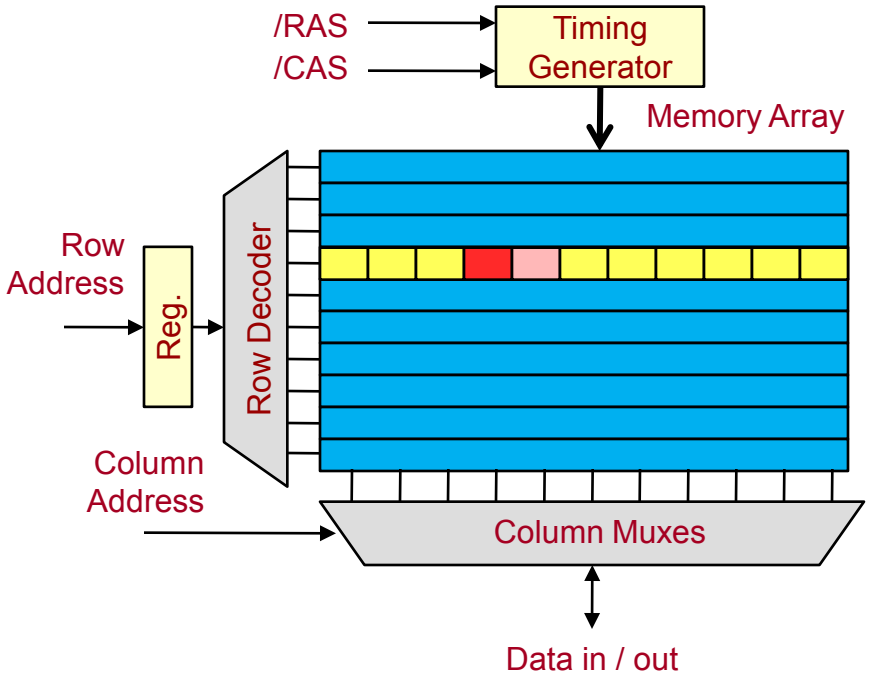
t_{RAC}=Access Time (60ns) = Time until data is valid

Legacy DRAM
(Must present new Row/Column address for each access)

Fast Page Mode DRAM Timing

- Can provide multiple column addresses with only one row address

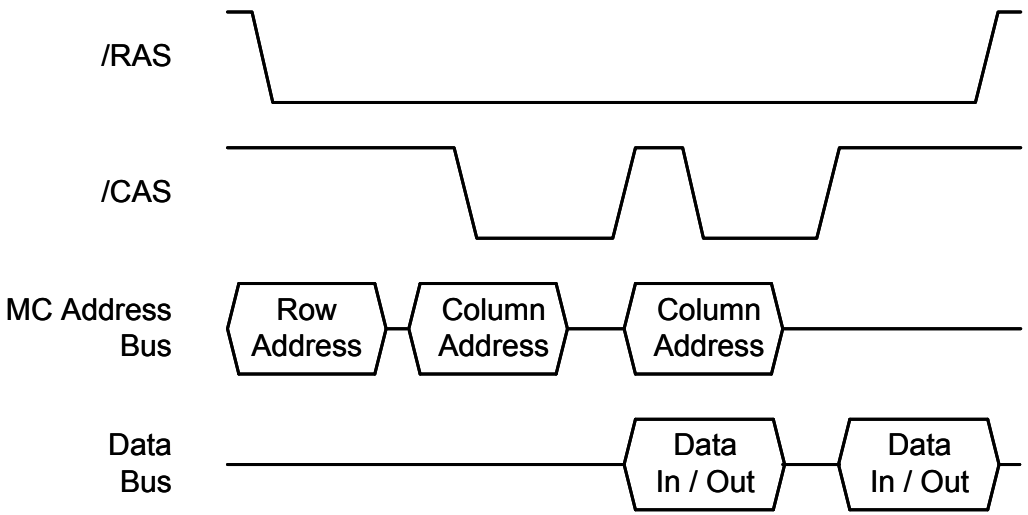


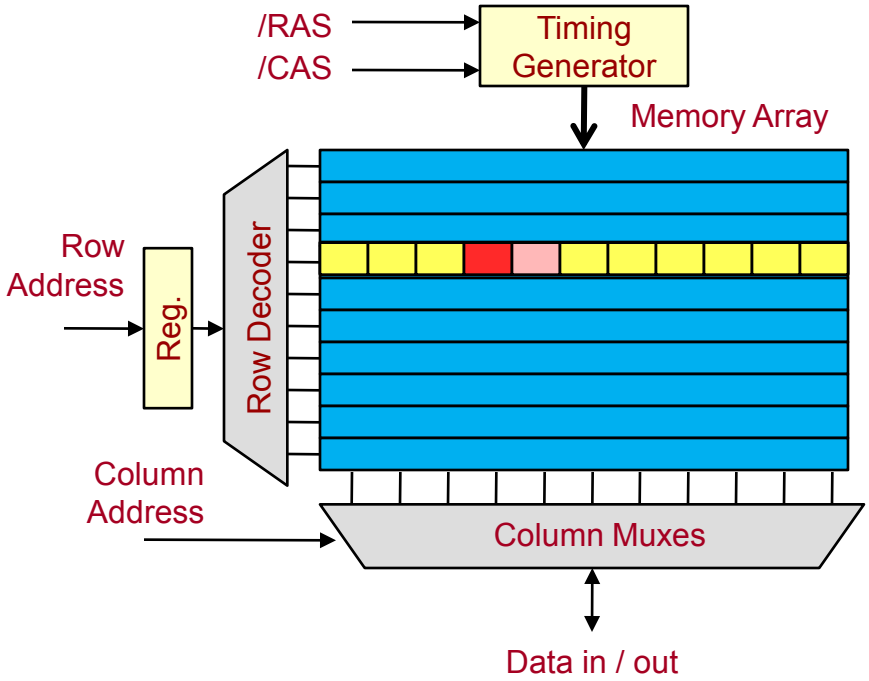


Fast Page Mode
(Future address that fall in same row can pull data from the latched row)

EDO DRAM Timing

- Similar to FPM but overlaps data i with column address i+1

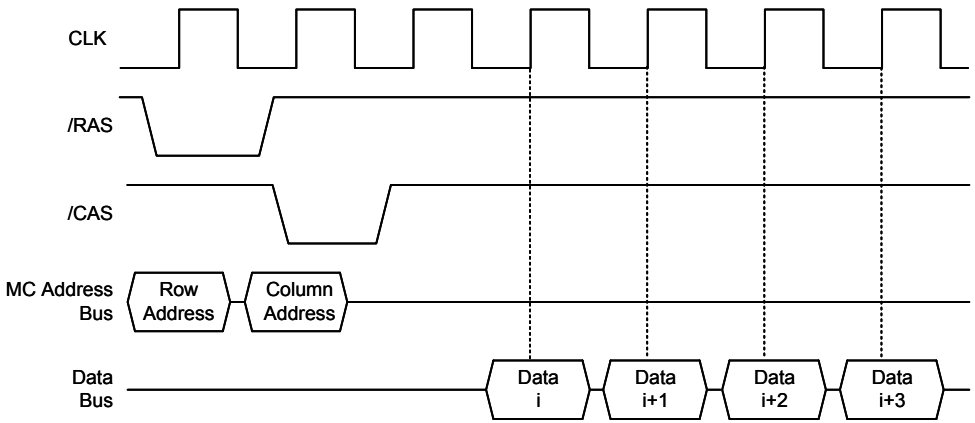


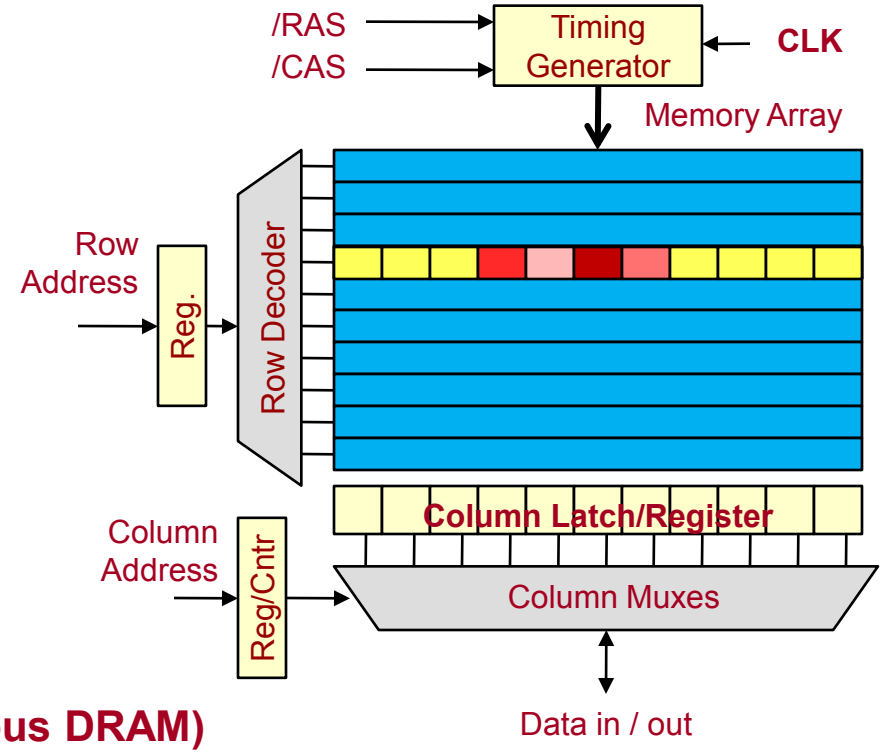


EDO (Extended Data Out)
Column address i+1 is sent while data i is being transferred on the bus

Synchronous DRAM Timing

- Registers the column address and automatically increments it, accessing *n* sequential data words in *n* successive clocks called bursts… n=4 or 8 usually)

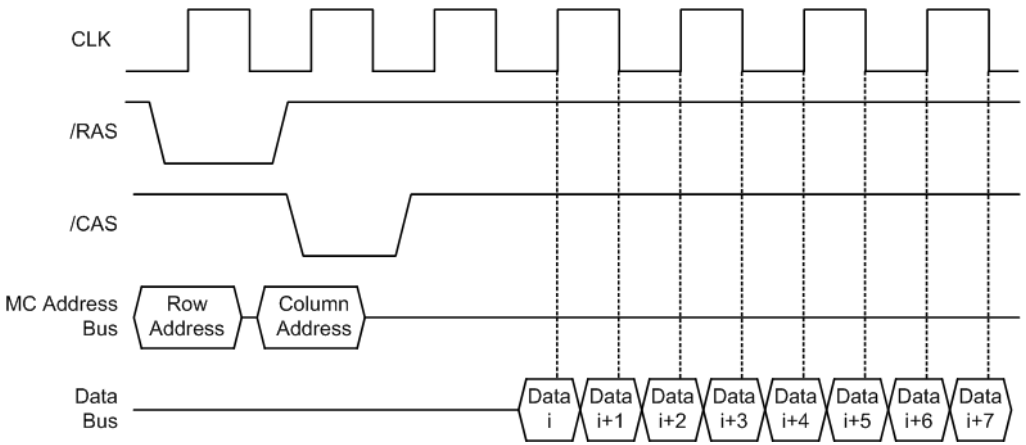


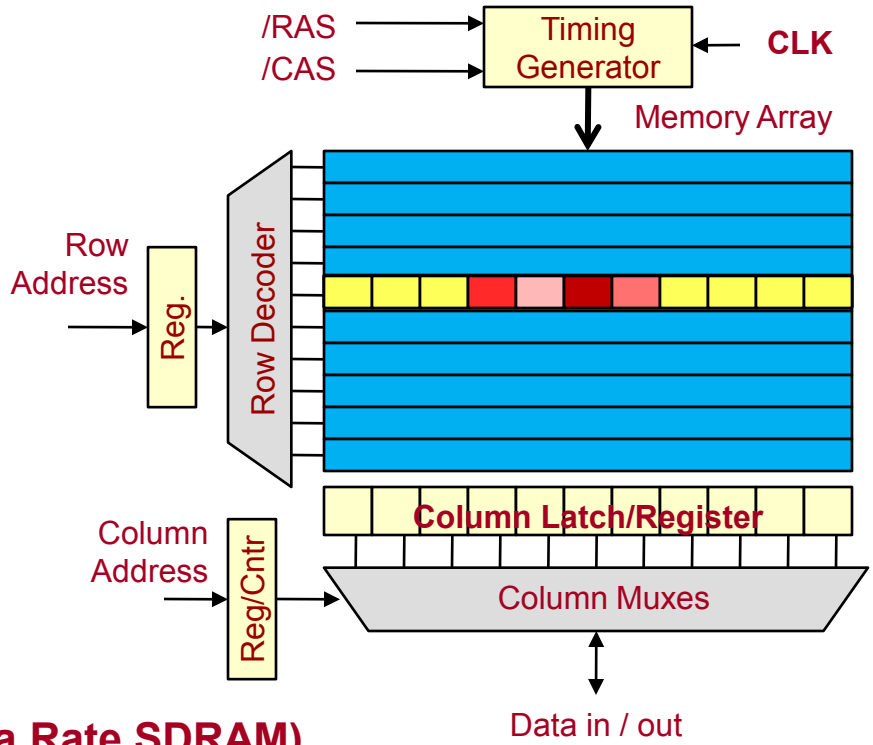


SDRAM (Synchronous DRAM)
Addition of clock signal. Will get up to 'n' consecutive words in the next 'n' clocks after column address is sent

DDR SDRAM Timing

- Double data rate access data every half clock cycle

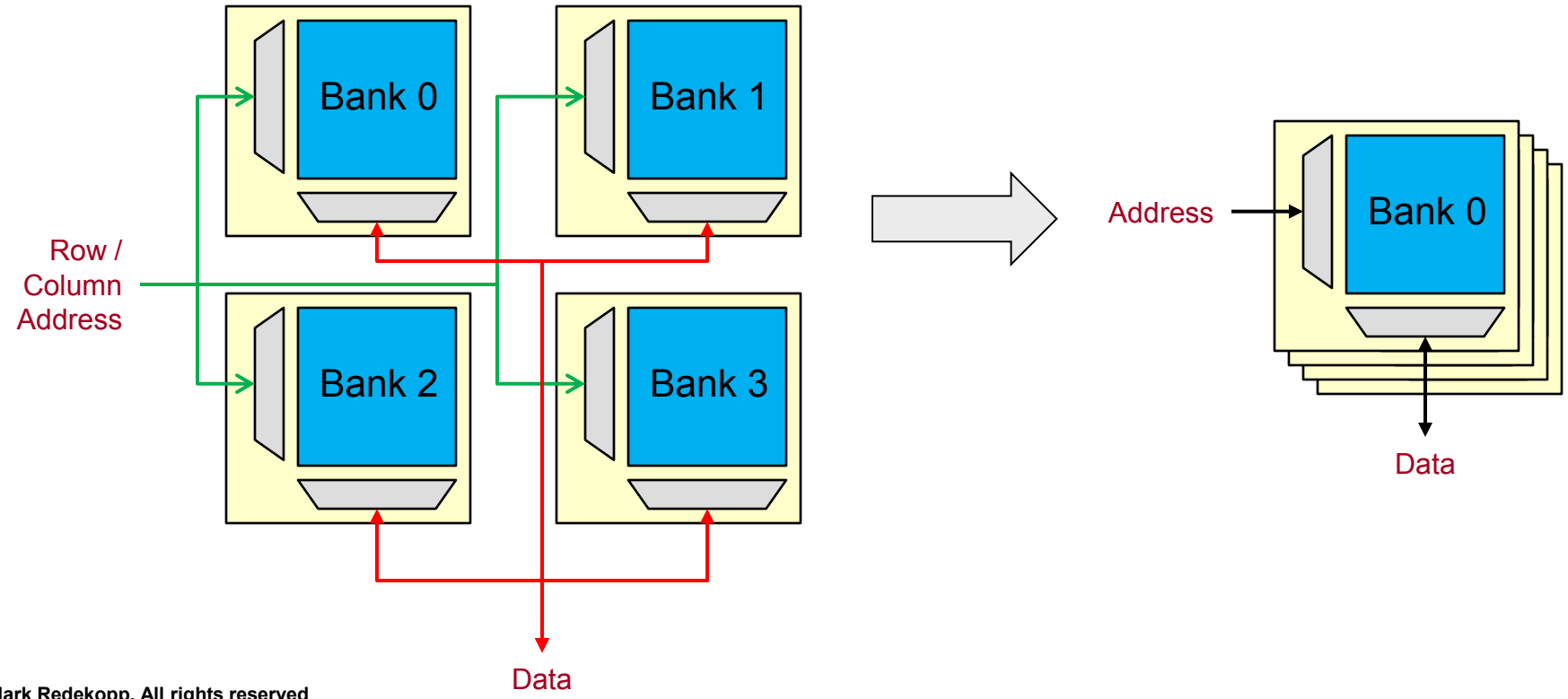




DDR SDRAM (Double-Data Rate SDRAM)
Addition of clock signal. Will get up to '2n' consecutive words in the next 'n' clocks after column address is sent

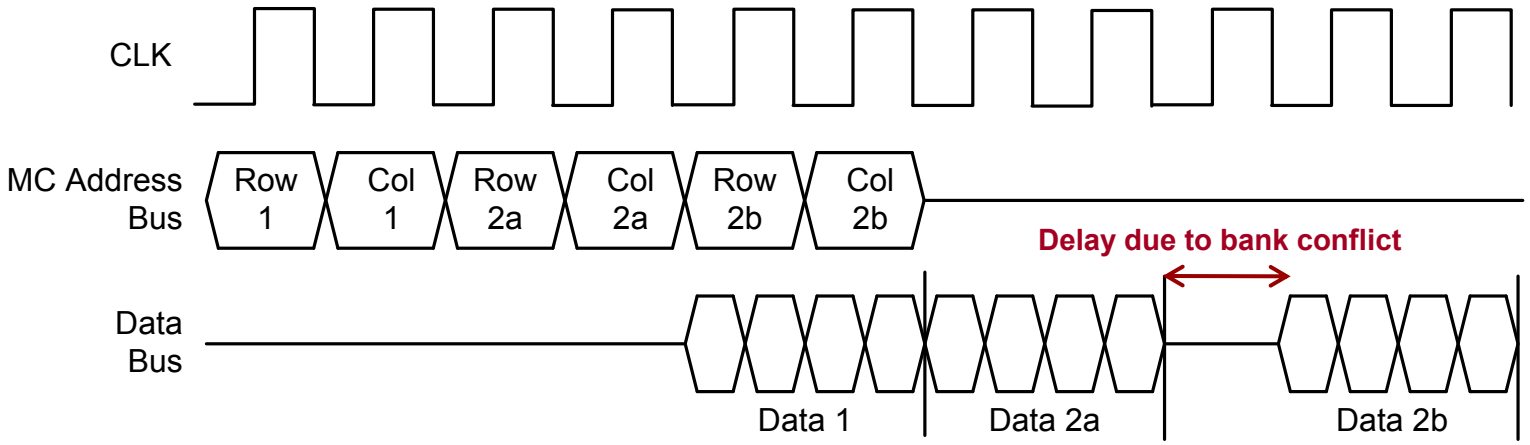
Banking

- Divide memory into “banks” duplicating row/column decoder and other peripheral logic to create independent memory arrays that can access data in parallel
  - uses a portion of the address to determine which bank to access



Bank Access Timing

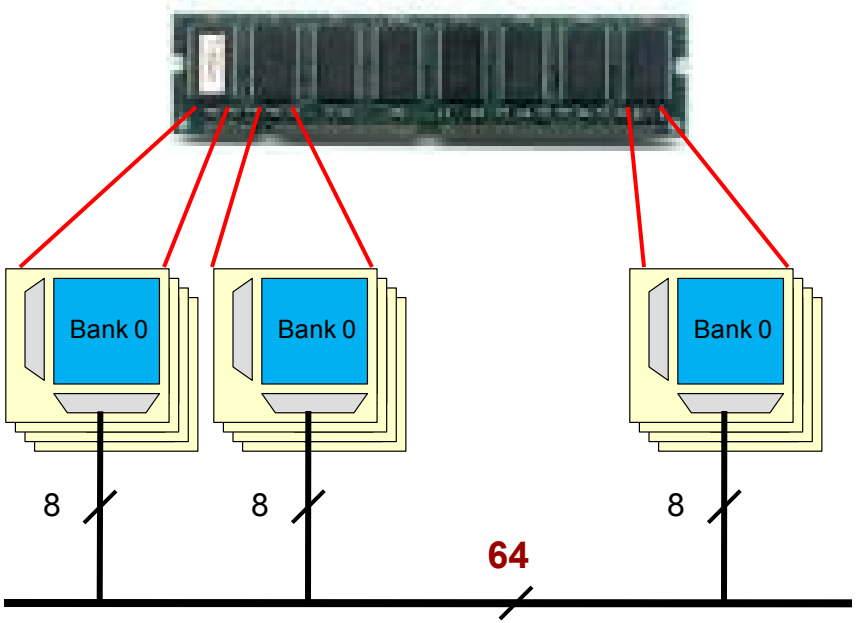
- Consecutive accesses to different banks can be overlapped and hide the time to access the row and select the column
- Consecutive accesses within a bank (to different rows) exposes the access latency



Access 1 maps to bank 1 while access 2a maps to bank 2 allowing parallel access. However, access 2b immediately follows and maps to bank 2 causing a delay.

Memory Modules

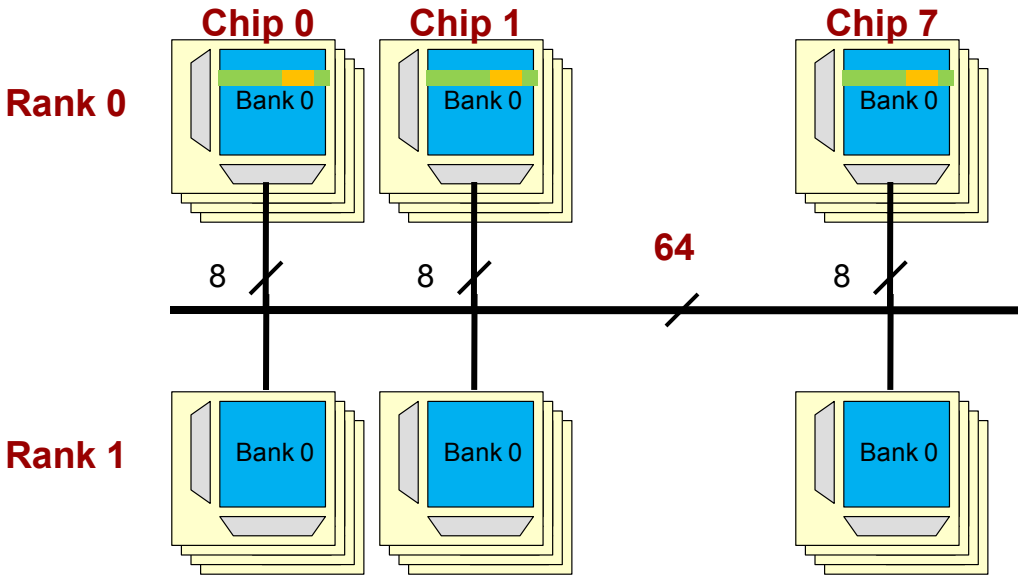
- Integrate several chips on a module
  - SIMM (Single In-line Memory Module) = single sided
  - DIMM (Dual In-line Memory Module) = double sided
- 1 side is termed a "rank"
  - SIMM = 1 rank
  - DIMM = 2 ranks
- Example
  - (8) 1Mx8 chips each output 8 bits (1-byte) to form a 64-bit (8-byte) data bus



Address Breakdown

- Assume a 1GB memory system made of 2 ranks of 64Mx8 chips (4 x 16K x 1K x 8) connected to a 64-bit (8-byte) wide data bus

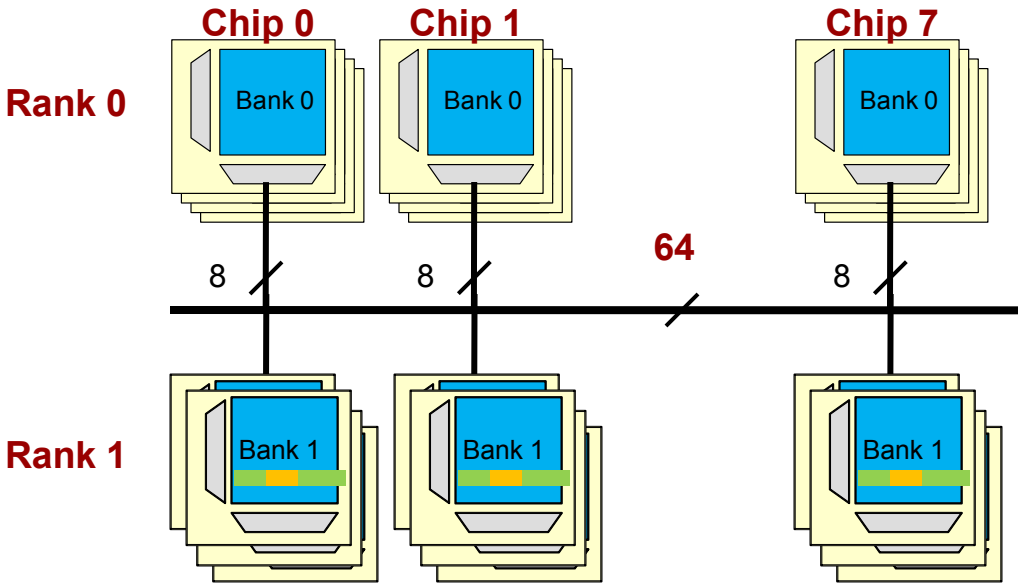
	Unused	Rank	Row	Bank	Col.	Unused
	A31,A30	A29	A28…A15	A14,A13	A12…A3	A2..A0
0x004f1ad8	00	0	00 0000 1001 1110	00	1101011011	000



Address Breakdown

- Assume a 1GB memory system made of 2 ranks of 64Mx8 chips (4 x 16K x 1K x 8) connected to a 64-bit (8-byte) wide data bus

	Unused	Rank	Row	Bank	Col.	Unused
	A31,A30	A29	A28…A15	A14,A13	A12…A3	A2..A0
0xff05aac0	11	1	11 1110 0000 1011	01	0101011000	000



Address Breakdown

- Assume 2 GB of memory on a single SIMM module with (8) 256MB each (broken into 8 banks and 8K rows)

Unused	Rank	Row	Bank	Col.	Unused
A31	-	A30-A18	A17-A15	A14-A3	A2-A0

Memory Summary

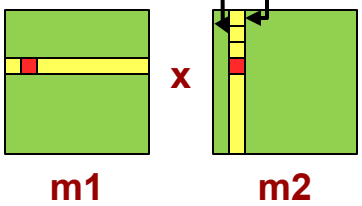
- Opportunities for speedup / parallelism
  - Sequential accesses lowers latency (due to bursts of data via FPM, EDO, SDRAM techniques)
  - Ensure accesses map to different banks and ranks to hide latency (parallel decoding and operation)

Programming Considerations

- For memory configuration given earlier, accesses to the same bank but different row occur on an 32KB boundary
- Now consider a matrix multiply of 8K x 8K integer matrices (i.e. 32KB x 32KB)
- In code below…m2[0][0] @ 0x10010000 while m2[1][0] @ 0x10018000

Unused	Rank	Row	Bank	Col.	Unused
A31,A30	A29	A28…A15	A14,A13	A12…A3	A2..A0
00	0	1 0000 0000 0001 0	00	0000000000	000
00	0	1 0000 0000 0001 1	00	0000000000	000

0x10010000
0x10018000

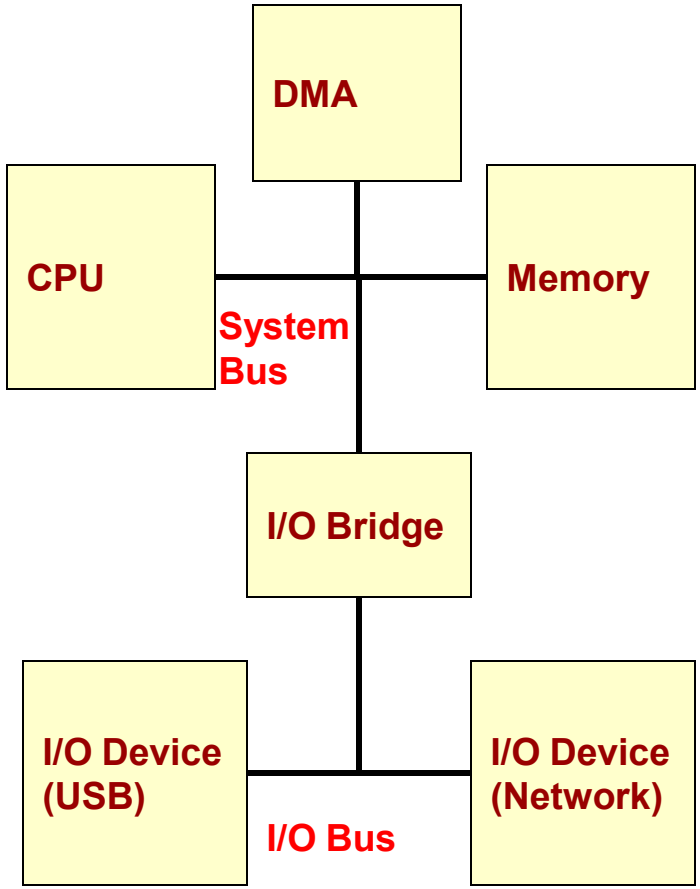



```
int m1[8192][8192], m2[8192][8192], result[8192][8192];
int i,j,k;
...
for(i=0; i < 8192; i++){
  for(j=0; j < 8192; j++){
    result[i][j]=0;
    for(k=0; k < 8192; k++){
      result[i][j] += matrix1[i][k] * matrix2[k][j];
} } }
```

DMA & ENDIAN-NESS

Direct Memory Access (DMA)

- Large buffers of data often need to be copied between:
  - Memory and I/O (video data, network traffic, etc.)
  - Memory and Memory (OS space to user app. space)
- DMA devices are small hardware devices that copy data from a source to destination freeing the processor to do "real" work

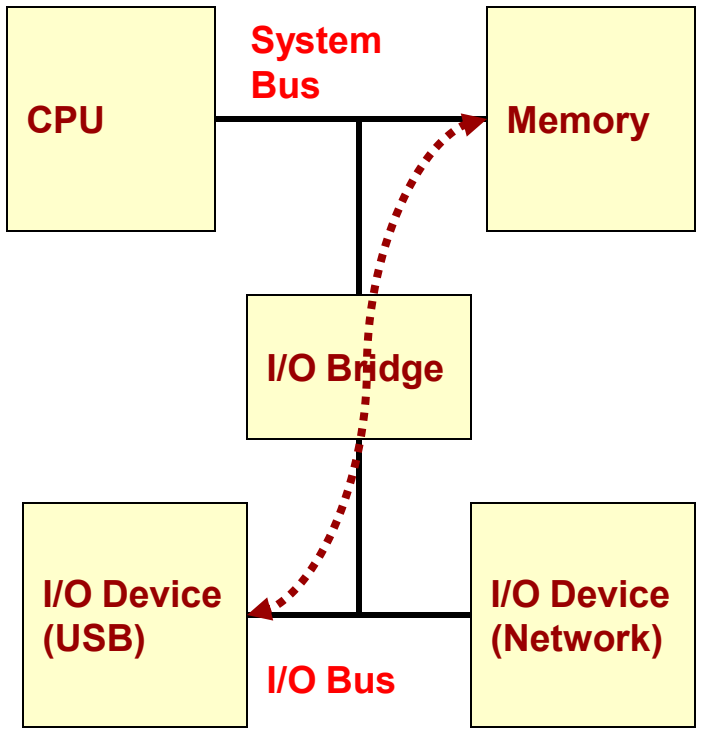


Data Transfer w/o DMA

- Without DMA, processor would have to move data using a loop
- Move 16Kwords pointed to by ($8) to ($9)

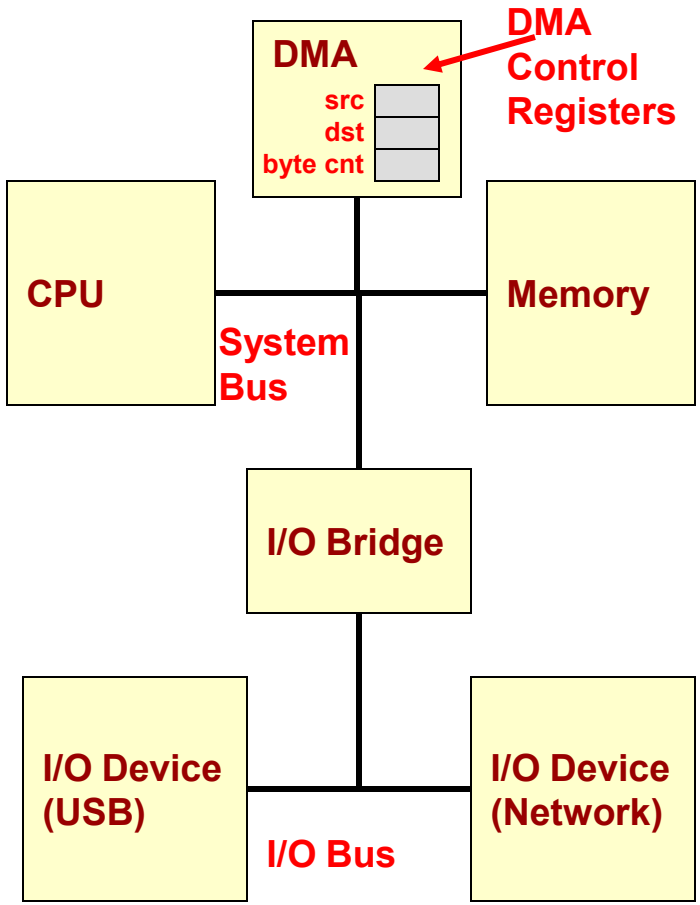
```
        addi   $18,$0,16384
AGAIN:  lw     $17,0($8)
        sw     $17,0($9)
        addi   $8,$8,4
        addi   $9,$9,4
        addi   $18,$18,-1
        bne    $18,$zero,AGAIN
```

- Processor wastes valuable execution time moving data



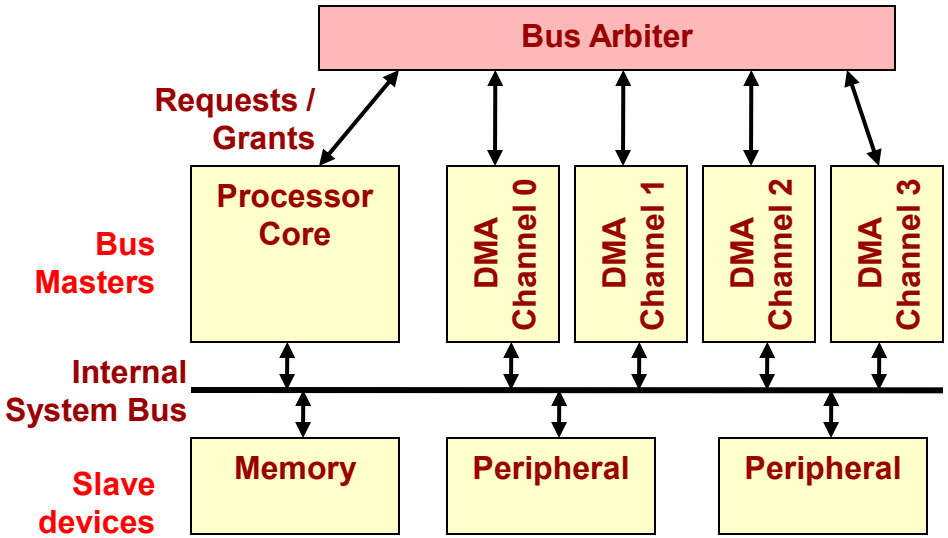
Data Transfer w/ DMA

- Processor sets values in DMA control registers
  - Source Start Address
  - Dest. Start Address
  - Byte Count
  - Control & Status (Start, Stop, Interrupt on Completion, etc.)
- DMA becomes "bus-master" (controls system bus to generate reads and writes) while processor is free to execute other code
  - Small problem: Bus will be busy
  - Hopefully, data & code needed by the CPU will reside in the processor's cache



DMA Engines

- Systems usually have multiple DMA engines/channels
- Each can be configured to be started/controlled by the processor or by certain I/O peripherals
  - Network or other peripherals can initiate DMA's on their behalf
- Bus arbiter assigns control of the bus
  - Usually winning requestor has control of the bus until it relinquishes it (turns off its request signal)



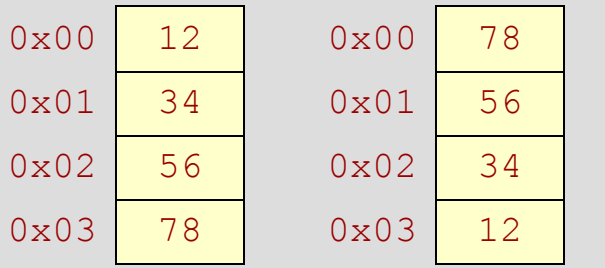
Endian-ness

- **Endian-ness** refers to the two alternate methods of ordering the **bytes** in a larger unit (word, long, etc.)
  - Big-Endian
    - PPC, Sparc
    - *MS byte* is put at the starting address
  - Little-Endian
    - used by Intel processors / PCI bus
    - *LS byte* is put at the starting address

The longword value:

0 x 1 2 3 4 5 6 7 8

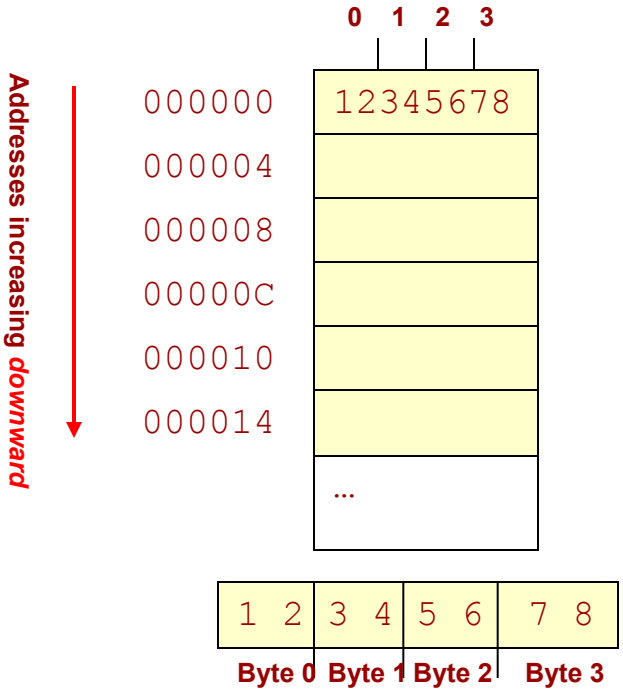
can be stored differently

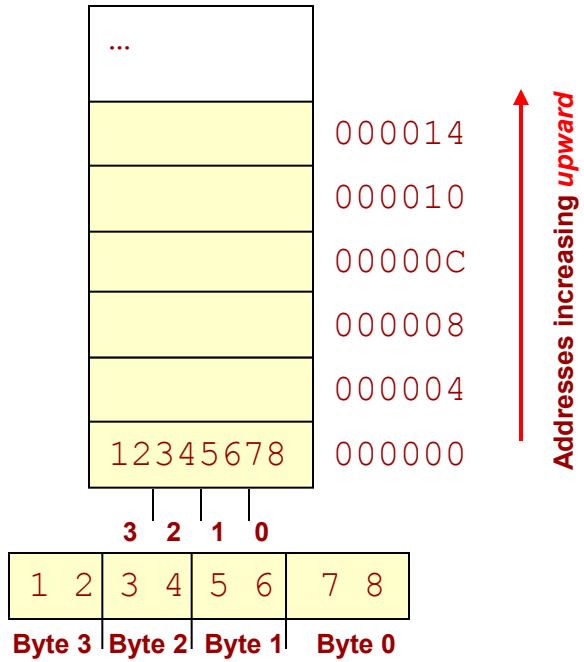


Big-Endian **Little-Endian**

Big-endian vs. Little-endian

- Big-endian
  - makes sense if you view your memory as starting at the top-left and addresses increasing as you go down
- Little-endian
  - makes sense if you view your memory as starting at the bottom-right and addresses increasing as you go up





Big-endian vs. Little-endian

- Issues arise when transferring data between different systems
  - Byte-wise copy of data from big-endian system to little-endian system
  - Major issue in networks (little-endian computer => big-endian computer) and even within a single computer (System memory => Peripheral (PCI) device)