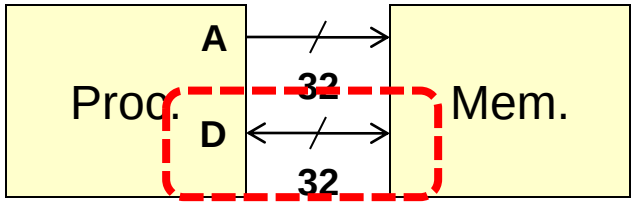
EE 457 Unit 7b

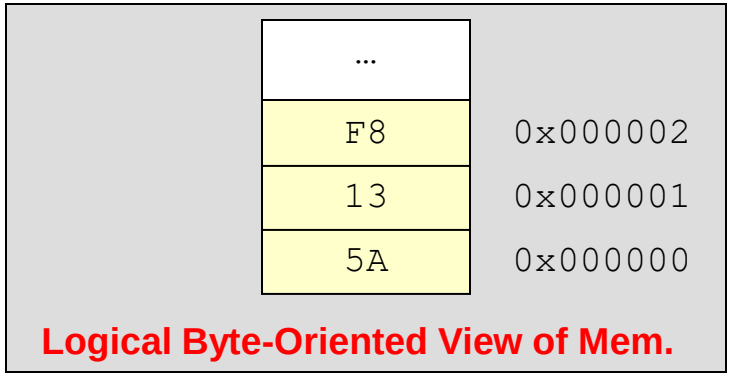
Main Memory Organization

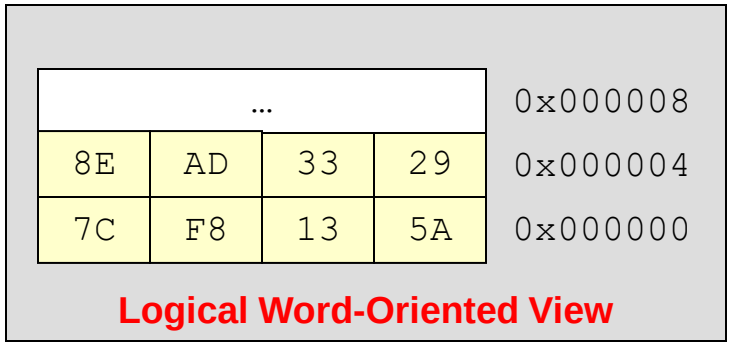
PROC/MEM PHYSICAL INTERFACE

Recall: MIPS Memory Data Organization

- We can logically picture memory in the units (sizes) that we actually access them
- We can access 1-byte at a time but the data bus allows for wider access (32-bits)
- Logical view of memory arranged in rows of largest access size (word)
  - Still with separate addresses for each byte
  - Can get word, halfwords, or bytes





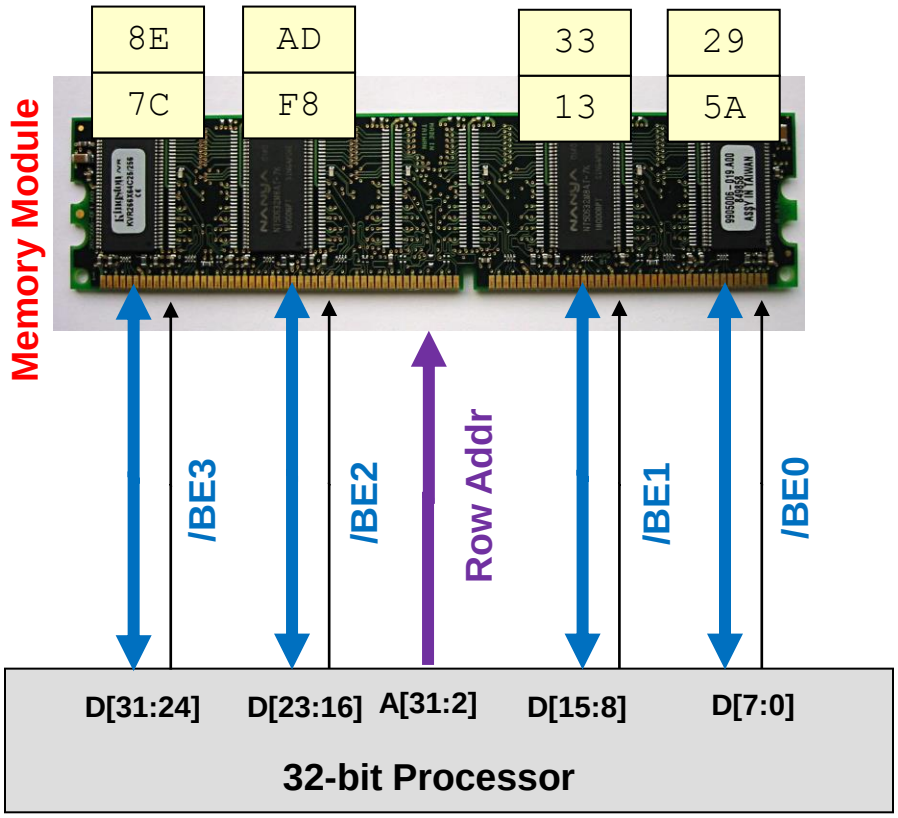


Byte Enables and the Data Bus

...				
7 8E	6 AD	5 33	4 29	0x000008
3 7C	2 F8	1 13	0 5A	0x000004
				0x000000

Logical Word-Oriented View

...				
1,3 8E	1,2 AD	1,1 33	1,0 29	0x000002
0,3 7C	0,2 F8	0,1 13	0,0 5A	0x000001
				0x000000
B3	B2	B1	B0	ROW



This Photo by Unknown Author is licensed under CC BY-SA-NC

Byte Enables and the Data Bus

- What are the control signals that indicate access size?
- Though we may have a 32-bit address bus A[31:0], physically the processor will convert the lower 2 address bits A[1:0] and the size information into 4 separate ***bank ("lane") enables [/BE3../BE0]***

Lane 3 /BE3	Lane 2 /BE2	Lane 1 /BE1	Lane 0 /BE0

Byte @ 40000003	Byte @ 40000002	Byte @ 40000001	Byte @ 40000000

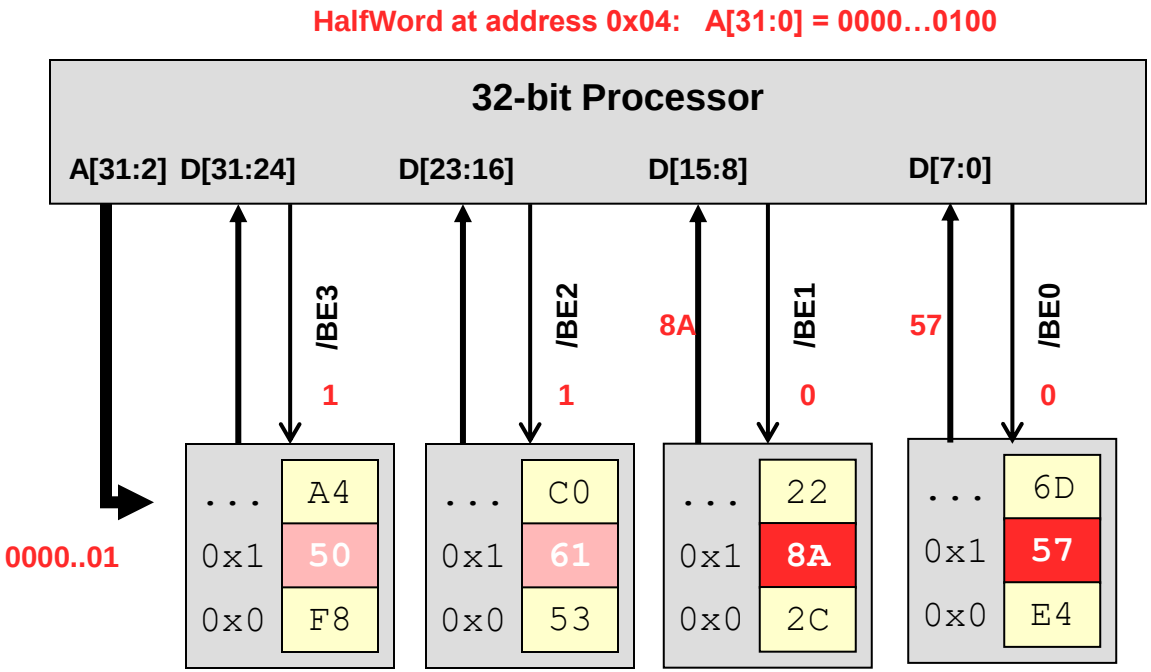
Halfword @ 40000002 | Halfword @ 4000000

Word @ 40000000

Desired Memory	Internal A[31:0]	Physical Addr. Bus A[31:2]	/BE3	/BE2	/BE1	/BE0
Word @ 0x4000000	0100…0000	0100…00	0	0	0	0
Half @ 0x40000002	0100…0010	0100…00	0	0	1	1
Byte @ 0x40000002	0100…0010	0100…00	1	0	1	1
Byte @ 0x40000001	0100…0001	0100…00	1	1	0	1
Half @ 0x40000004	0100…0100	0100…01	1	1	0	0

Address & Data Bus Connections

- Organize memory into several byte-size memories running in parallel (sometimes known as “banks”)
- Convert lower address bits into bank enables to selectively enable each bank
- A[31:2] is provided to all memory banks specifying the same internal location



Byte Addressable Processors

Proc.	External Data Bus	Address Pin-Out	Min. # of Banks	Shift in Address
8088 (8-bit proc.)	D[7:0]	A[19:0]	1	0
8086 (16-bit proc.)	D[15:0]	A[19:1]	2	1
80386 (32-bit proc.)	D[31:0]	A[23:2]	4	2
Core Series (64-bit proc.)	D[63:0]	A[35:3]	8	3

MEMORY INTERLEAVING

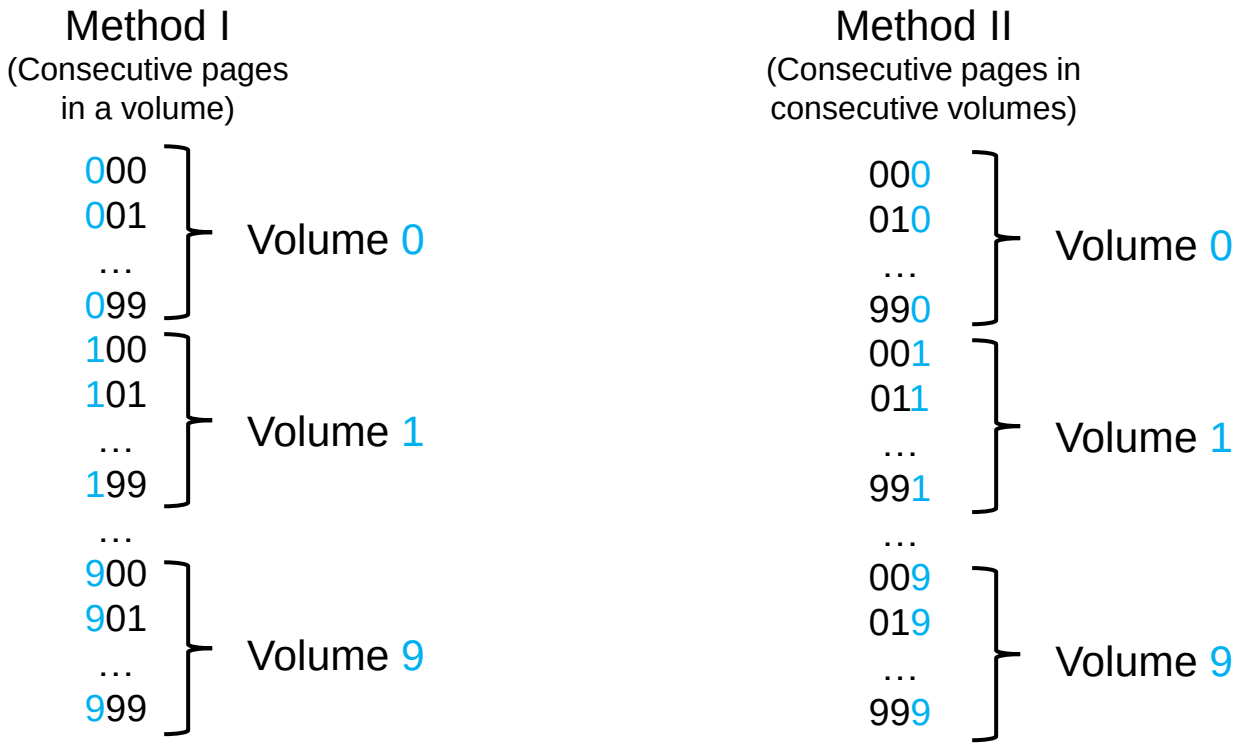
Motivation

- Organize main memory to
  - Facilitate byte-addressability

  while maintaining…
  - Efficient fetching of the words in a cache block
- Low order interleaving (L.O.I) helps us achieve this

Interleaving Analogy

- Consider a journal consisting of 1000 pages (000-999) bound in
  - 10 volumes (0-9) of
  - 100 pages each (00-99)



Interleaving Analogy

- Example: Say article 73 runs from page 730-739
  - In Method I: Article 73 is completely in volume 7
  - In Method II: The 73rd page of each volume form article 73 as shown below
- Which do you prefer?
  - If reading the article you may say method I
  - If you have to make a copy of the article and you have 10 photocopy machines with 10 friends to help you might say method II
    - Back to the scenario of reading the article, given those same 10 friends they could open each volume to page 73 for you so that you can read in a continuous manner

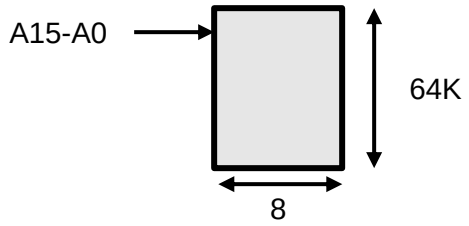
Page 730 is page 73 of volume 0
Page 731 is page 73 of volume 1
…
Page 739 is page 73 of volume 9

Low Order Interleaving

Byte Addressability

1. Intel 8085: 16-bit addr., 8-bit data, byte addressable processor.

   Memory space: 2^{16} = 64KB, A15-A0, D7-D0

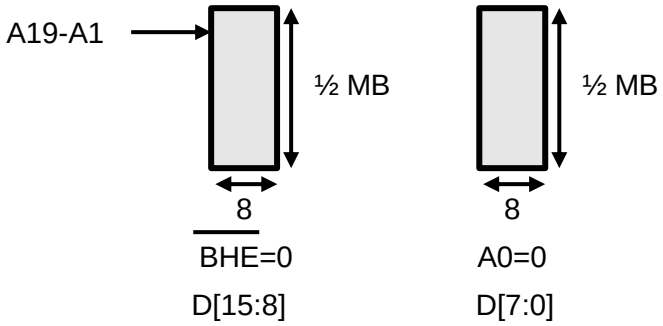


2. Intel 8086: 20-bit addr., 16-bit data, byte addressable, little-endian proc.

   Memory space: 2^{20} = 1MB, A19-A0

   [A19-A1, $\overline{BHE}$ ($\overline{BE1}$), A0 ($\overline{BE0}$)], D15-D0

   Byte 41 | Byte 40 = **Word 40**

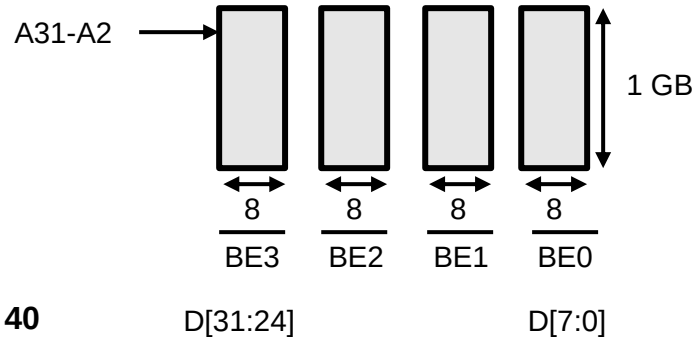


3. Intel 80386: 32-bit addr., 32-bit data, byte addressable, little-endian proc.

   Memory space: 2^{32} = 4GB, A31-A0

   [A31-A2, $\overline{BE3}$, $\overline{BE2}$, $\overline{BE1}$, $\overline{BE0}$], D31-D0

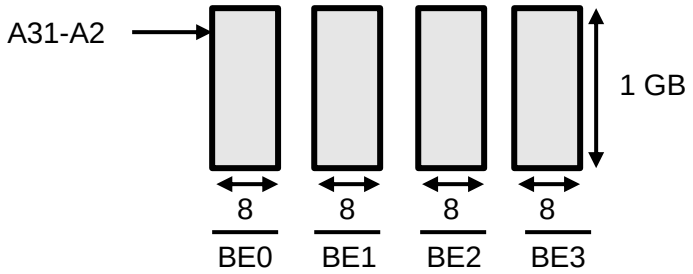
   Byte 43 | Byte 42 | Byte 41 | Byte 40 = **Word 40**



Byte Addressability

4. Intel 80386: 32-bit addr., 32-bit data, byte addressable, big-endian proc.

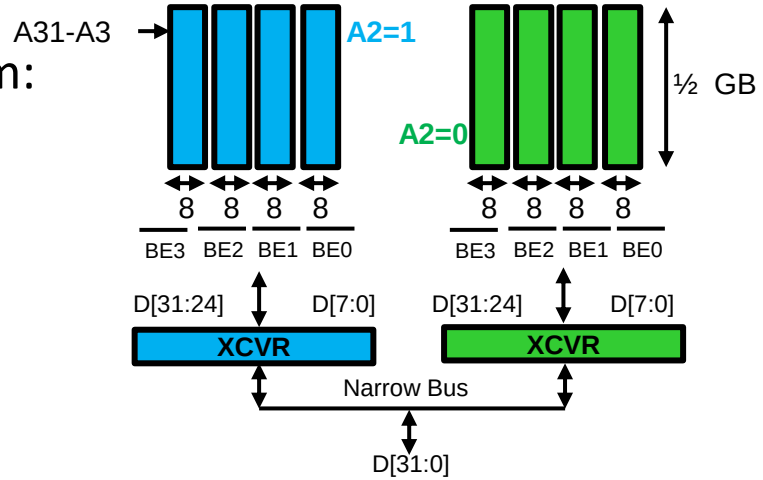
Memory space: 2^{32} = 4GB, A31-A0
[A31-A2, BE3, BE2, BE1, BE0], D31-D0



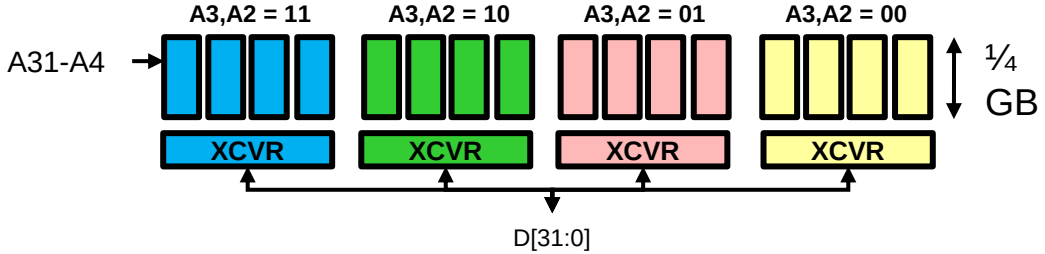
Byte 40	Byte 41	Byte 42	Byte 43	= Word 40

5. Little-Endian system, **2-way interleaved** system:
32-bit addr., 32-bit data,
byte addressable
(Narrow, 32-bit data bus b/w mem. and cache)

Memory space: 2^{32} = 4GB, A31-A0
[A31-A2, BE3, BE2, BE1, BE0], D31-D0

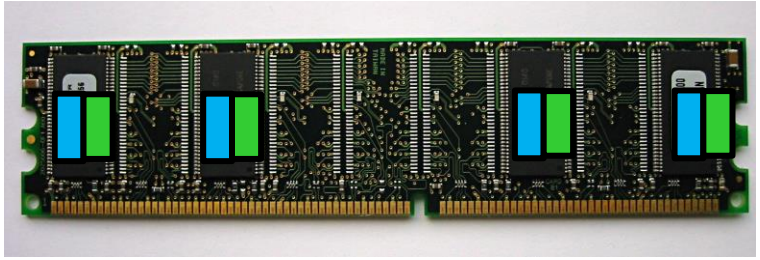


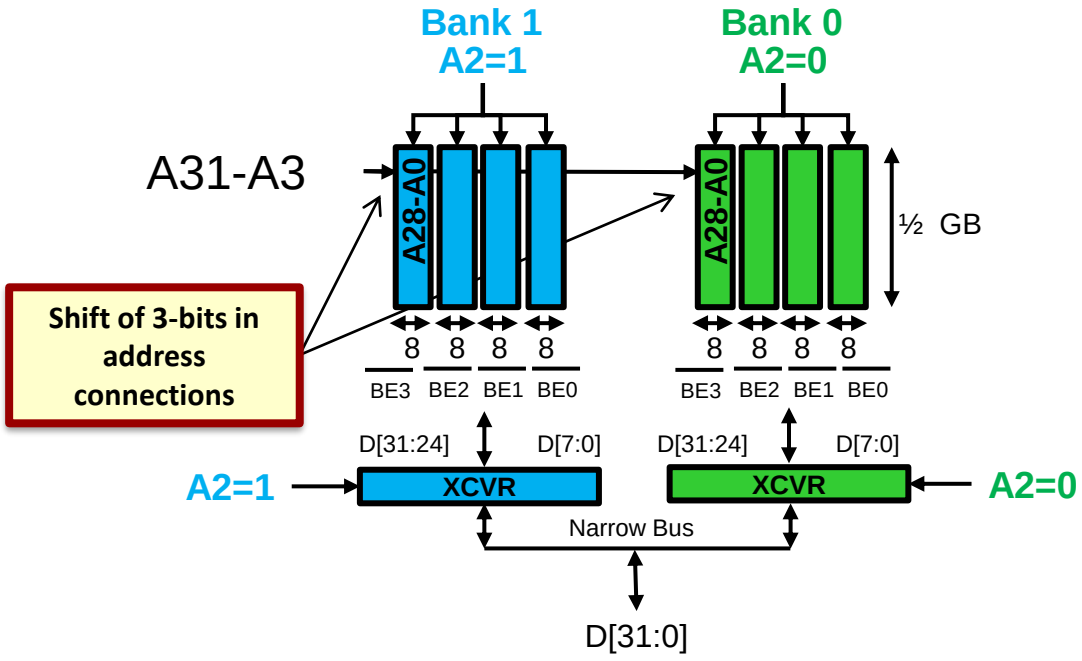
6. Same as 5 above,
but **4-way interleaved**



2-Way L.O.I.

- System address bus uses
  - A1:A0 and size info to generate /BE3../BE0 (Byte Enables)
    - In a 32-bit data bus, we need 2 address bits to produce the 4 BE's
    - In a 64-bit data bus, we would need 3 address bits to produce 8 BE's
  - Lower order bits to select a "bank"
    - Only 1 address bit, A2, to select one of 2 banks
  - Upper bits connect to each memory chip
    - Each memory chip is just a collection of ½ GB requiring 29 address bits…we can connect appropriate 29 bits

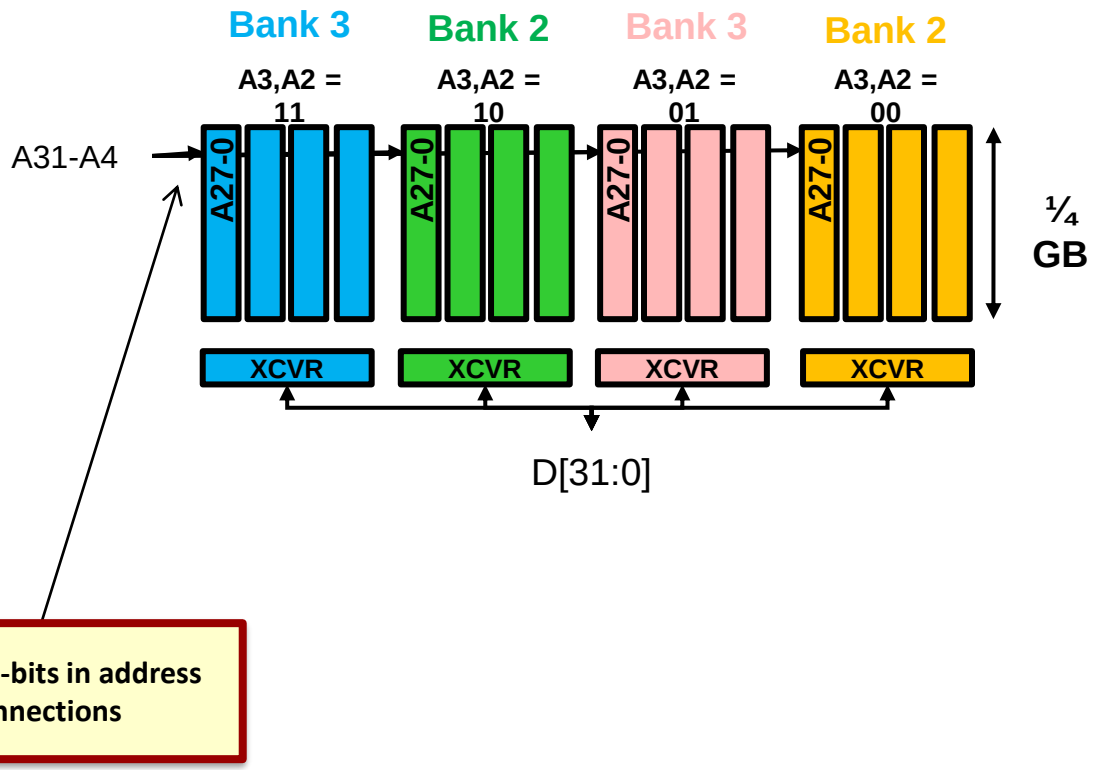




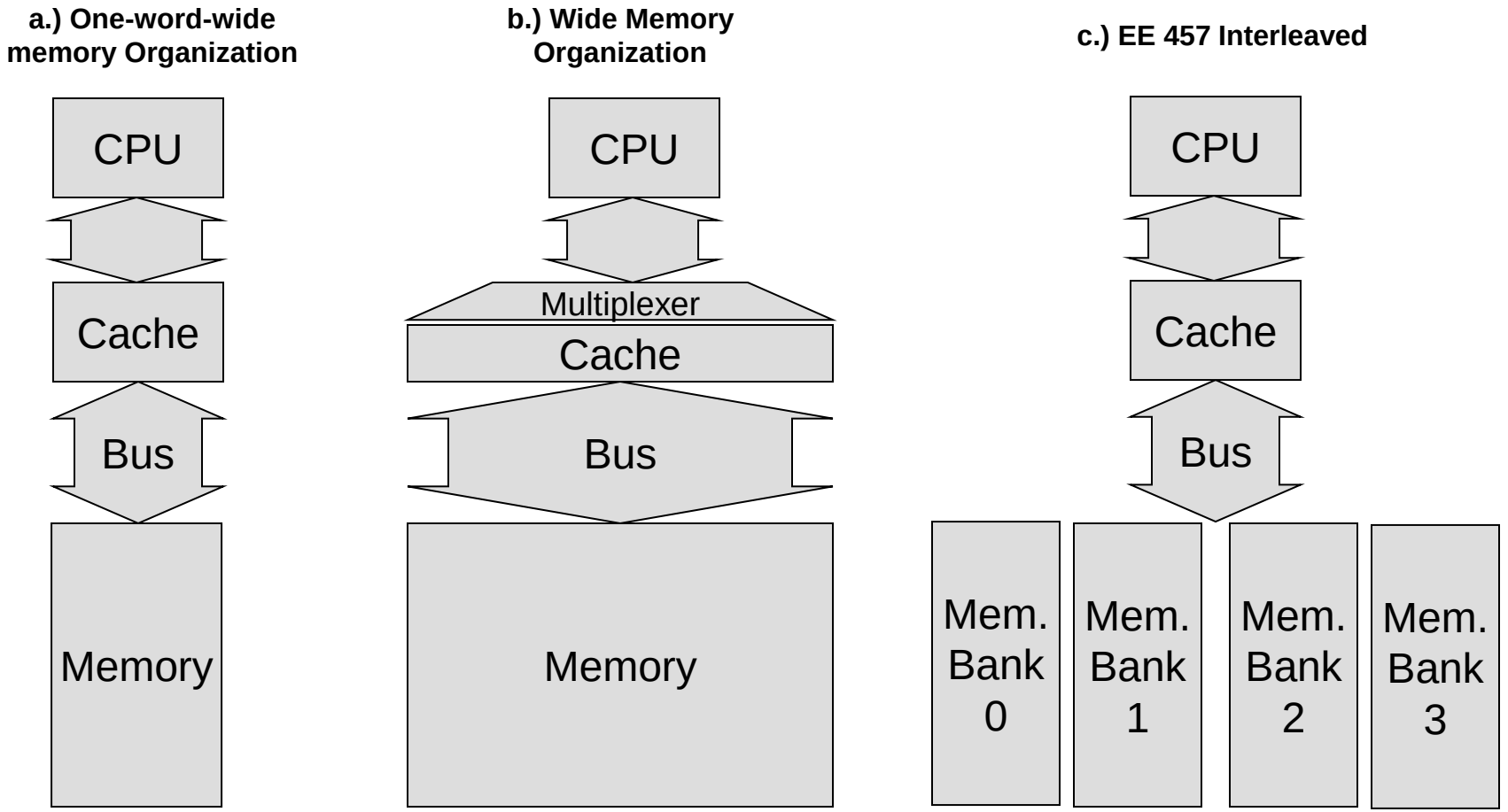
This Photo by Unknown Author is licensed under CC BY-SA-NC

4-Way L.O.I.

- System address bus uses
  - A1:A0 and size info to generate /BEi (Byte Enables)
  - Lower order bits to select a “bank”
  - Upper bits connect to each memory chip



Organization Options



Organization Comparison

- Assume following latencies

Send address to MM	1 clock
MM (DRAM) Access Time	15 clocks
Transfer time for one word	1 clock

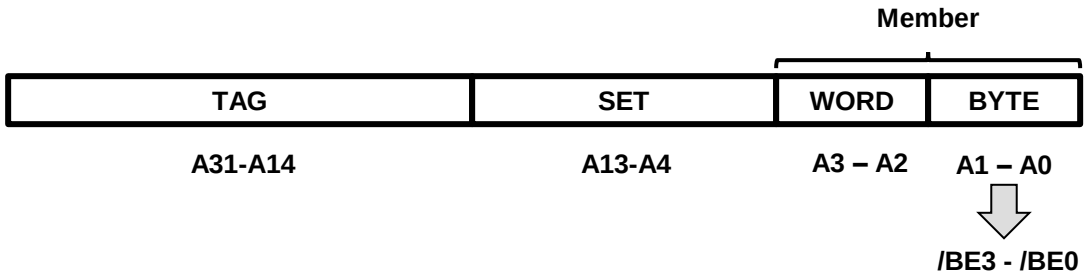
- Find time to access a cache line of 4-words

a. Narrow Memory	1 + 4*15 + 4*1 = 65 clocks (assume mem. controller will auto-increment address)
b. Wide Memory	1 + 15 + 1 = 17 clocks
c. Interleaved Memory	1 + 15 + 4*1 = 20 clocks

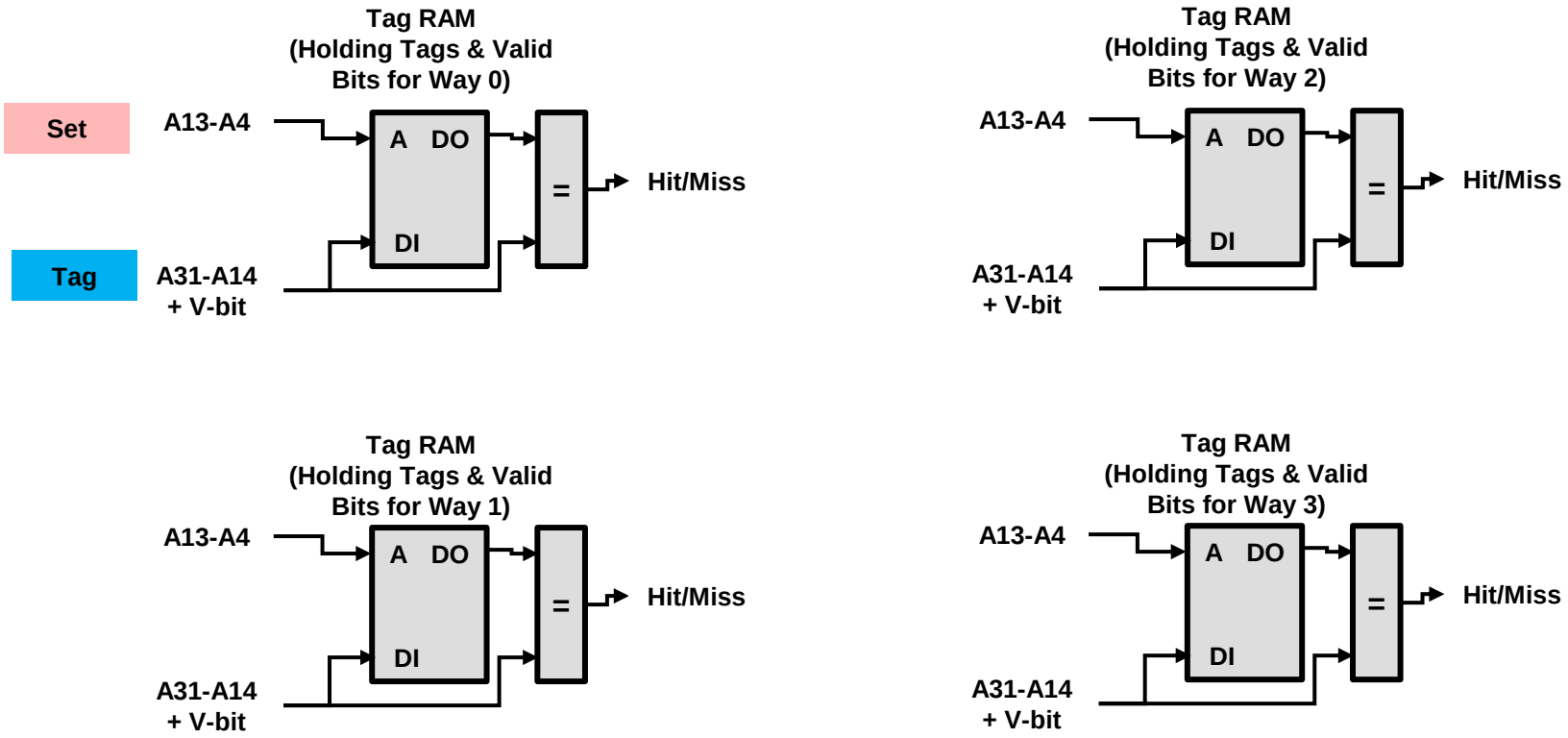
Example

- Consider a set-associative mapping and physical organization of main memory, cache data RAMs, and cache tag RAMs.
- Specs:
  - 32-bit physical address, byte-addressable system
  - Cache Size = 64KB
  - Block Size = 4 words (16 bytes)
  - Set Size = 4 blocks (64 bytes)

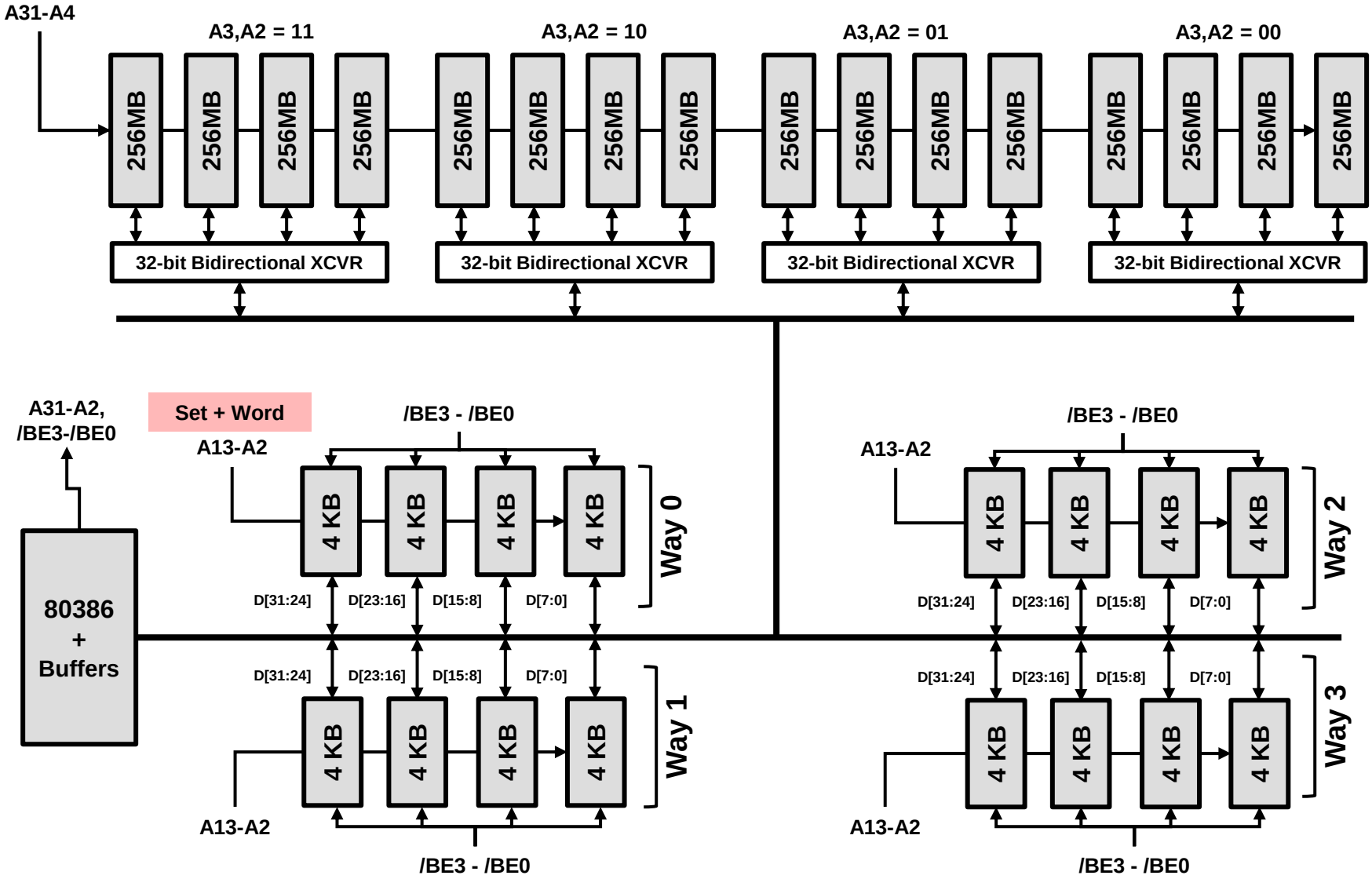
# of MM Blocks = 2^{32} / 2^4 = 2^{28}
# of Cache Blocks = 2^{16} / 2^4 = 2^{12}
# of Sets = 2^{12} cache blocks / 2^2 blocks/set = 2^{10}
# of Groups = 2^{28} MM blocks / 2^{10} sets = 2^{18}



Tag RAM Example



MM & Data RAM Example

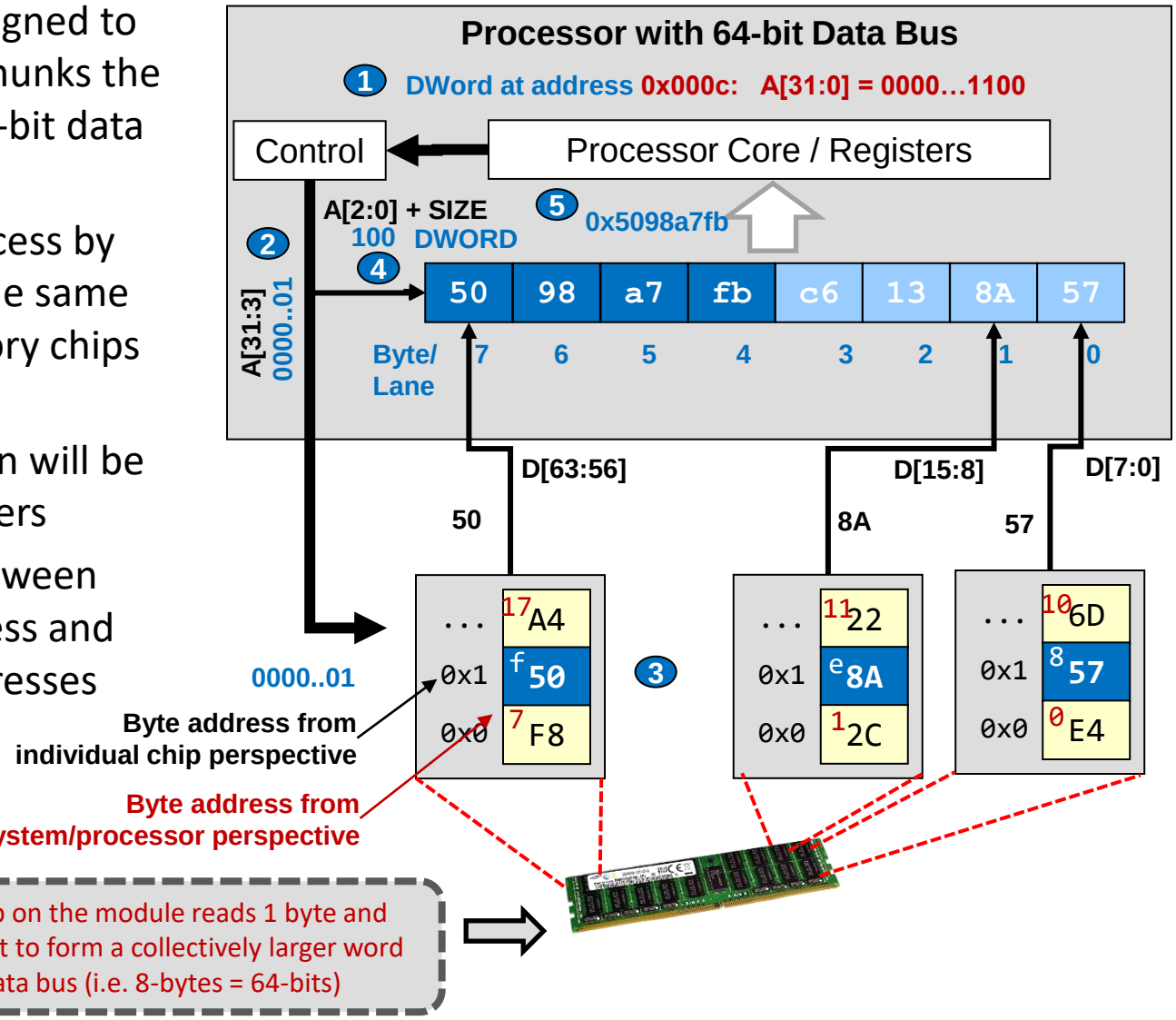


Main memory organization

DRAM TECHNOLOGIES

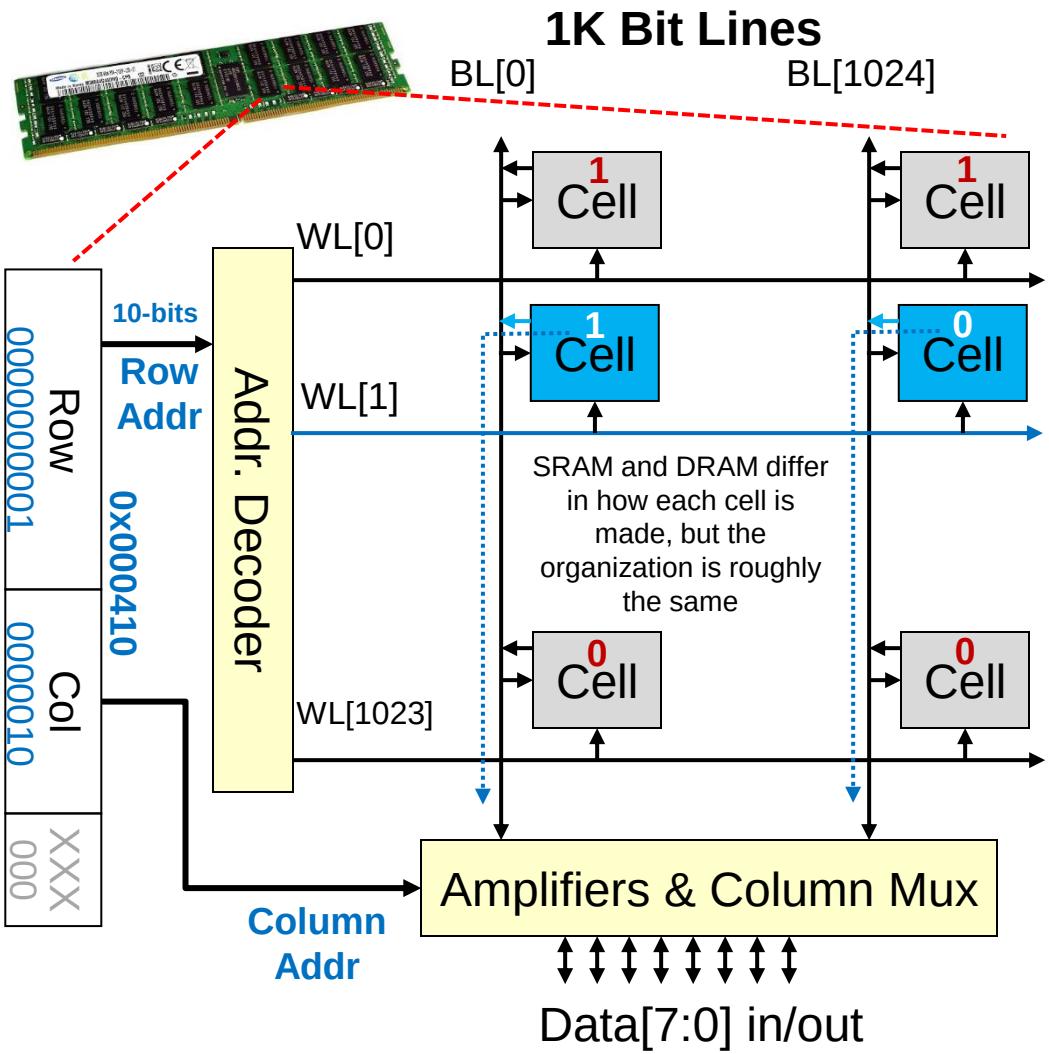
Memory Module Organization

- Memory module is designed to always access data in chunks the size of the data bus (64-bit data bus = 64-bit accesses)
- Parallelizes memory access by accessing the byte at the same location in all (8) memory chips at once
- Only the desired portion will be forwarded to the registers
- Note the difference between system processor address and local memory chip addresses



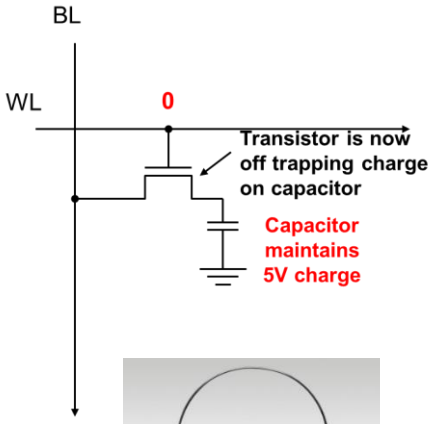
Memory Chip Organization

- Memory technologies share the same layout but differ in their cell implementation
  - SRAM
  - DRAM
- Memories require the row bits be sent first and are used to select one row (aka "word line")
  - Uses a hardware component known as a decoder
- All cells in the selected row access their data bits and output them on their respective "bit line"
- The column address is sent next and used to select the desired 8 bit lines (i.e. 1 byte)
  - Uses a hardware component known as a mux



SRAM vs. DRAM

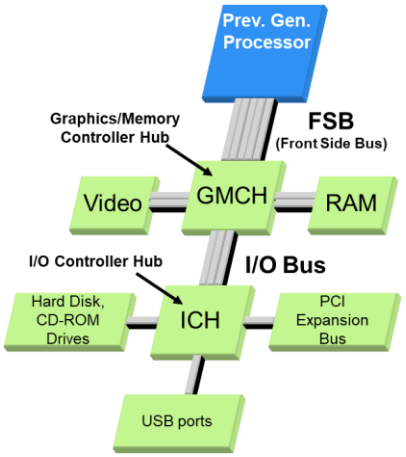
- Dynamic RAM (DRAM) Cells (store 1 bit)
  - Will lose values if not refreshed periodically every few milliseconds [i.e. dynamic]
  - Extremely small (1 Transistor & a capacitor)
    - Means we can have very high density (GB of RAM)
  - Small circuits require more time to access the bit
    - SLOW
  - Used for main memory
- Static RAM (SRAM) Cells (store 1 bit)
  - Will retain values as long as power is on [i.e. static]
  - Larger (6 transistors)
  - Larger circuitry can access bit faster
    - FASTER
  - Used for cache memory



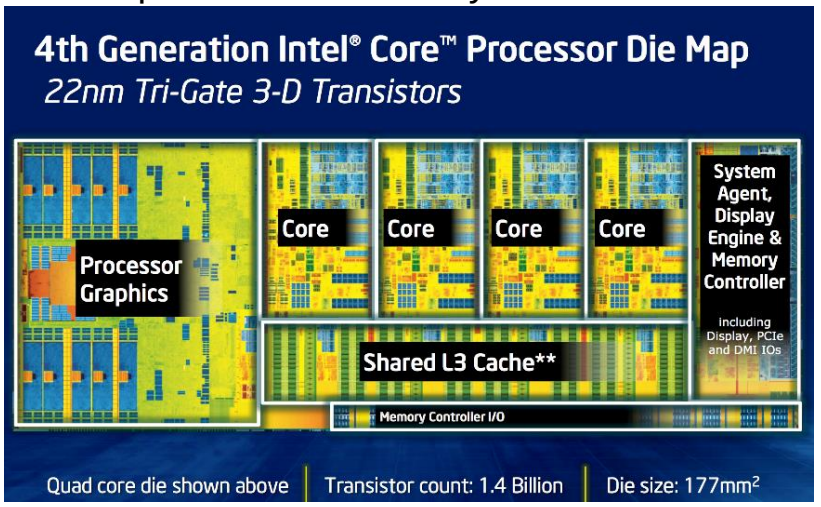
This Photo by Unknown Author is licensed under CC BY-NC

Memory Controller

- DRAMs require non-trivial hardware controller (aka memory controller)
  - To split up the address and send the row and column address as the right time
  - To periodically refresh the DRAM cells
  - Plus more…
- Used to require a separate chip from the processor
- But due to scaling (i.e. Moore's Law) most processors integrate the controller on-chip
  - Helps reduce access time since fewer hops



Legacy architectures used separate chipsets for the memory and I/O controller



Current general-purpose processors usually integrate the memory controller on chip.

Implications of Memory Technology

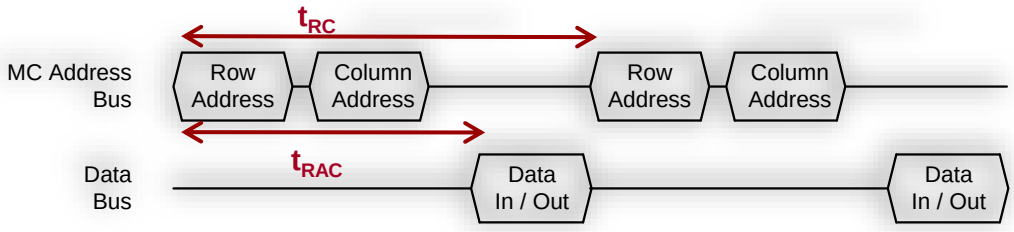
- Memory latency of a single access using current DRAM technology will be slow
- We must improve bandwidth
  - Idea 1: Access more than just a single word at a time (to exploit spatial locality)
  - Technology: Fast Page Mode, DDR SDRAM, etc.
  - Idea 2: Increase number of accesses serviced in parallel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

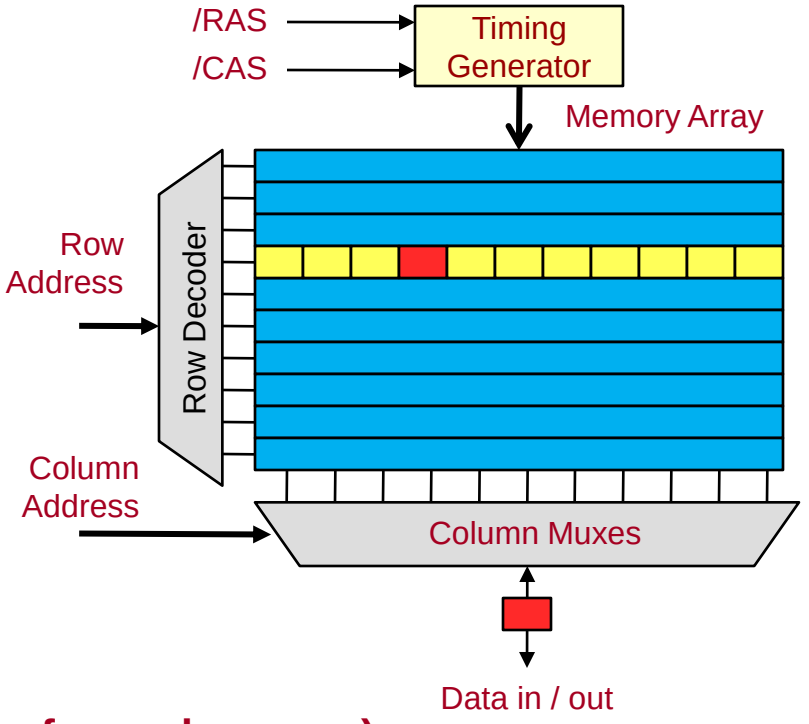
Legacy DRAM Timing

- Can have only a single access "in-flight" at once
- Memory controller must send row and column address portions for each access

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

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

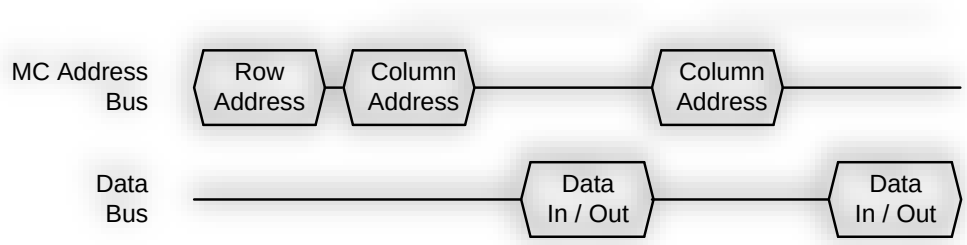


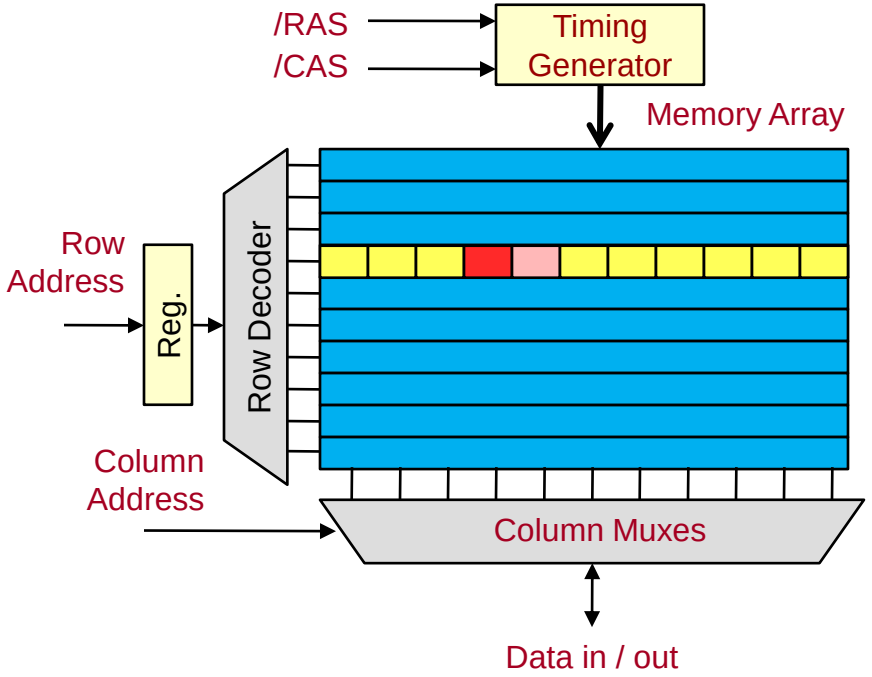


Legacy DRAM
(Must present new Row/Column address for each access)

Fast Page Mode DRAM Timing

- Can provide multiple column addresses with only one row address

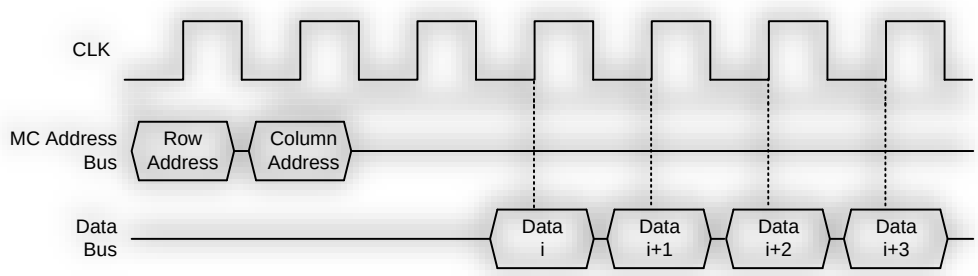


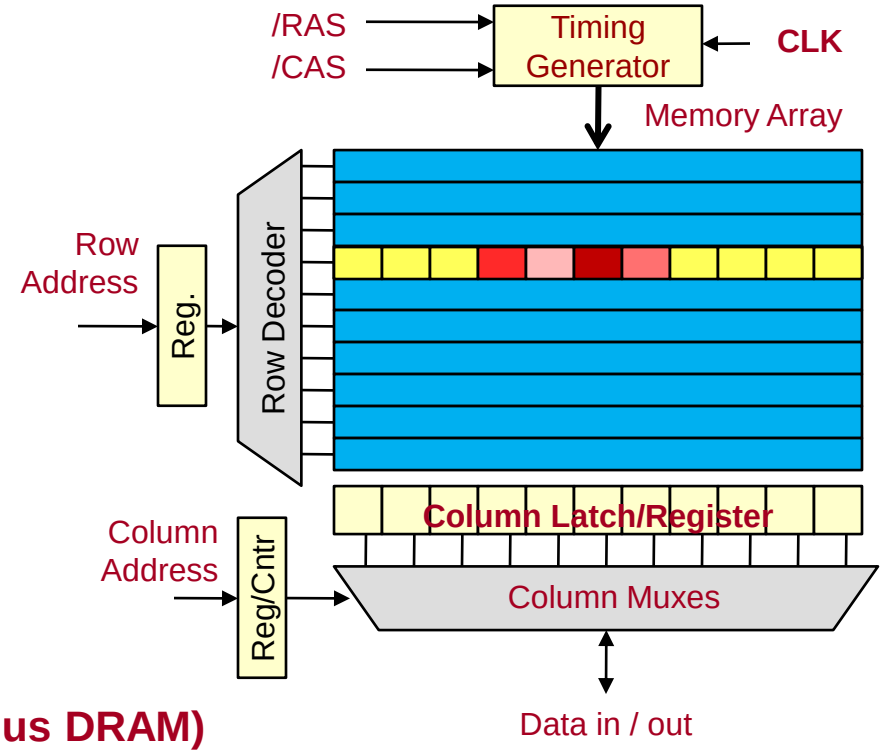


Fast Page Mode
(Future address that fall in same row can pull data from the latched row)

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)

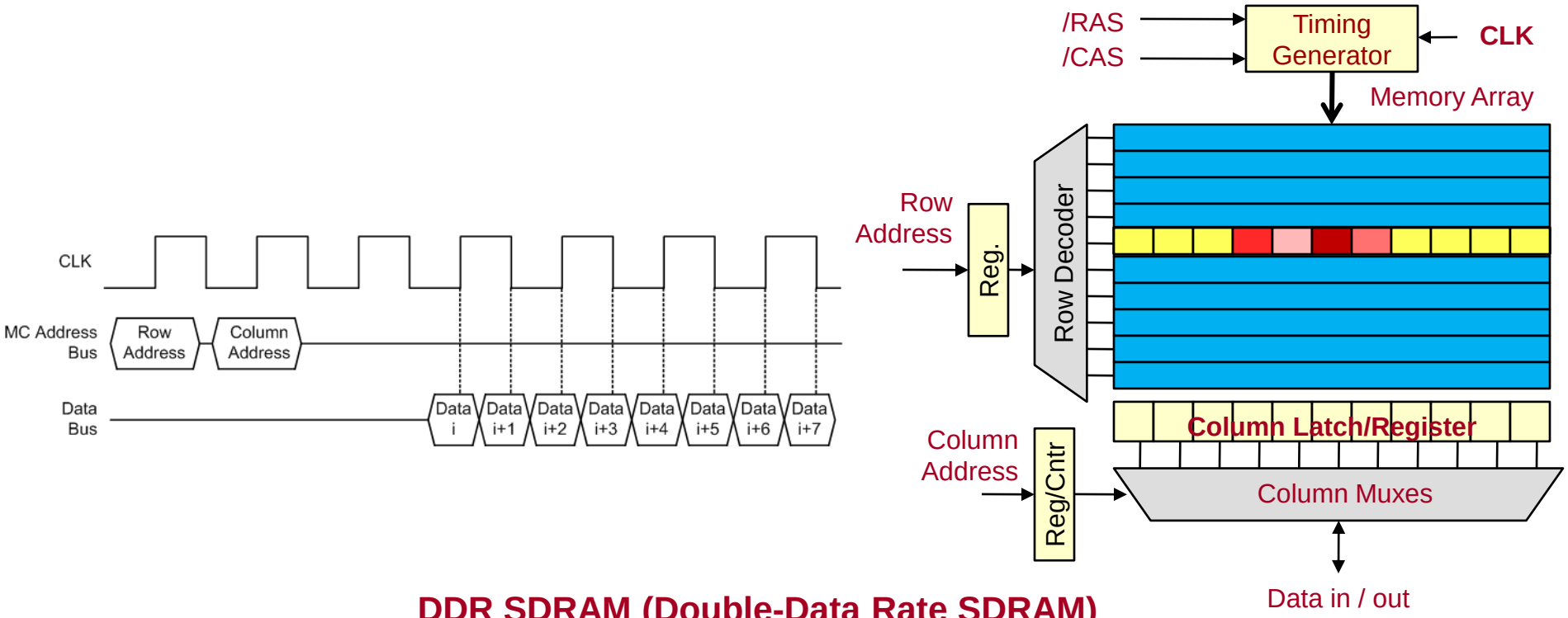




SDRAM (Synchronous DRAM)
Addition of clock signal. Will get up to 'n' consecutive words in the next 'n' clocks after column address is sent

DDR SDRAM Timing

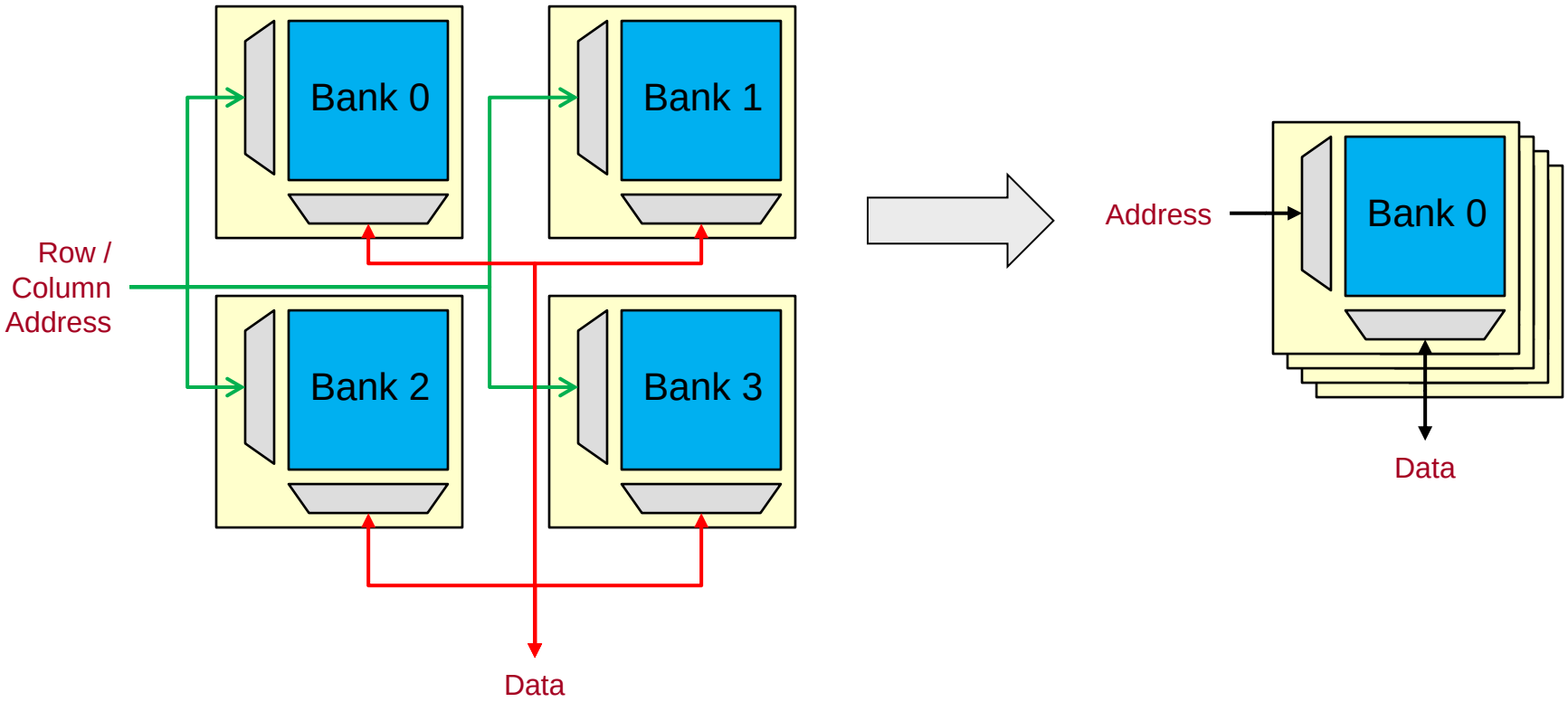
- Double data rate access data every half clock cycle



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

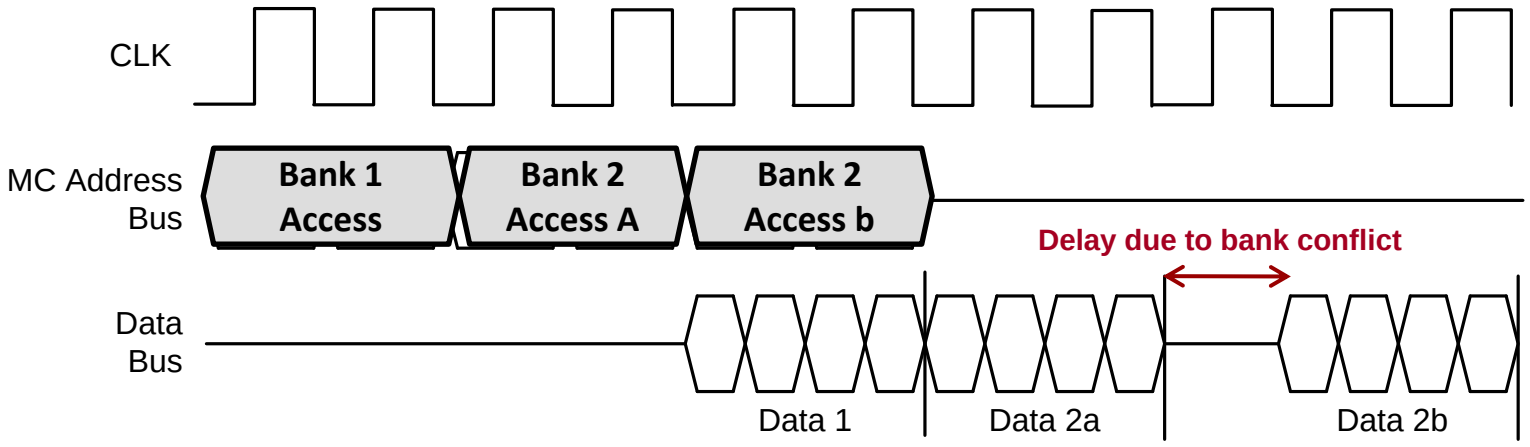
Banking

- 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

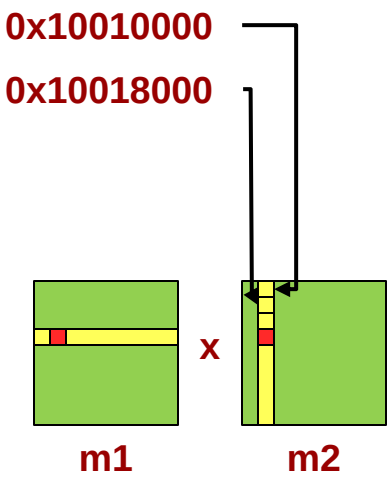
- Consecutive accesses to different banks can be overlapped and hide the time to access the row and select the column
- Consecutive accesses within a bank (to different rows) exposes the access latency



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.

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

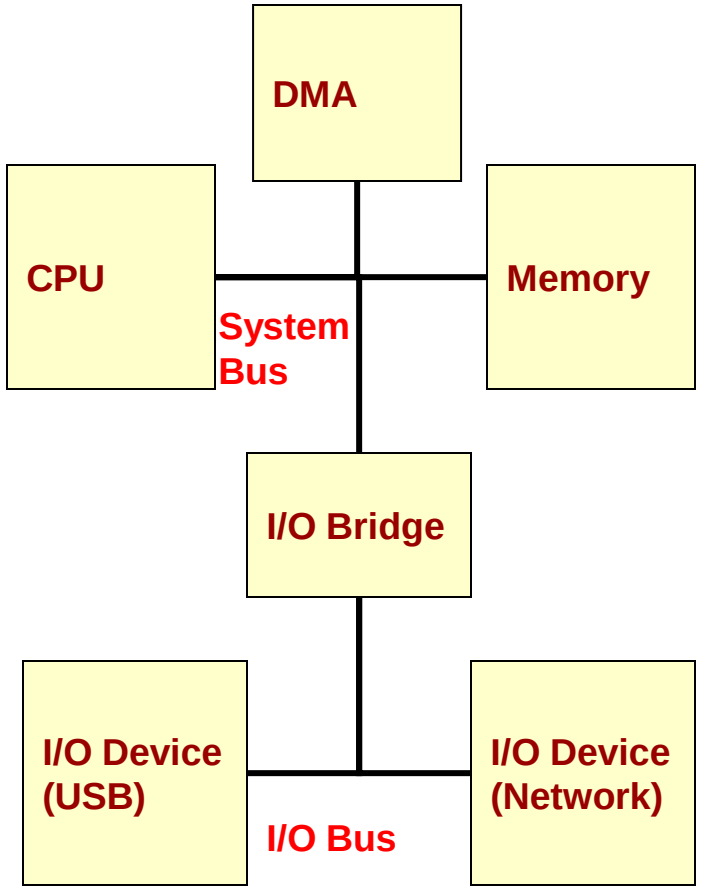
```
int m1[8192][8192], m2[8192][8192], result[8192][8192];
int i,j,k;
...
for(i=0; i < 8192; i++){
  for(j=0; j < 8192; j++){
    result[i][j]=0;
    for(k=0; k < 8192; k++){
      result[i][j] += matrix1[i][k] * matrix2[k][j];
} } }
```

Direct Memory Access

DMA

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

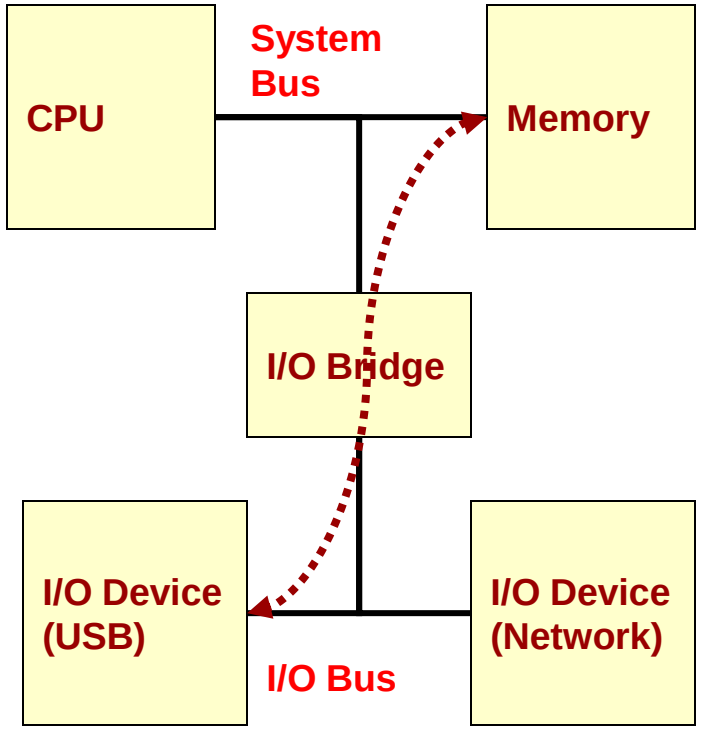


Data Transfer w/o DMA

- Without DMA, processor would have to move data using a loop
- Move 16Kwords pointed to by ($s1) to ($s2)

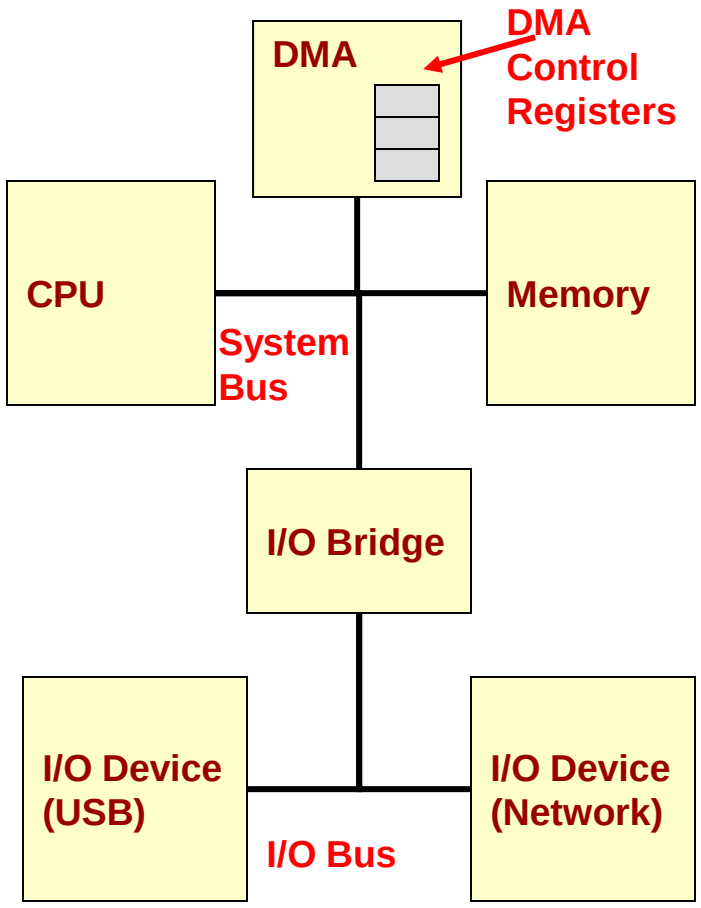
```
        li    $t0,16384
AGAIN:  lw    $t1,0($s1)
        sw    $t1,0($s2)
        addi  $s1,$s1,4
        addi  $s2,$s2,4
        subi  $t0,$t0,1
        bne   $t0,$zero,AGAIN
```

- 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)