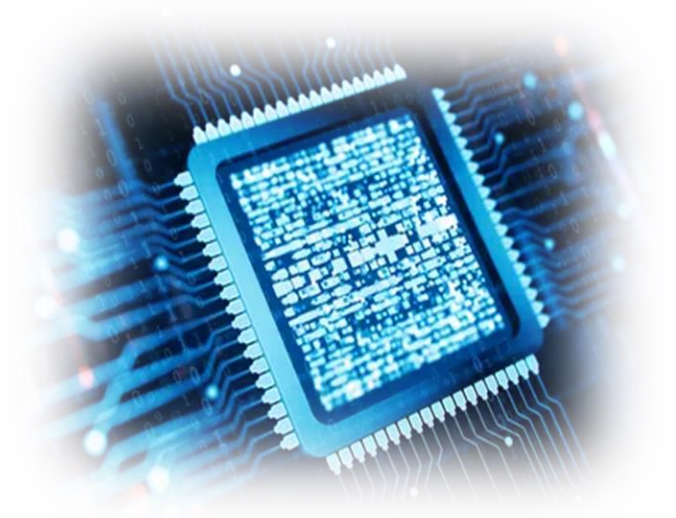


# MICROPROCESSOR SYSTEMS

BLG212E

Burak Berk Üstündağ CRN: 11450

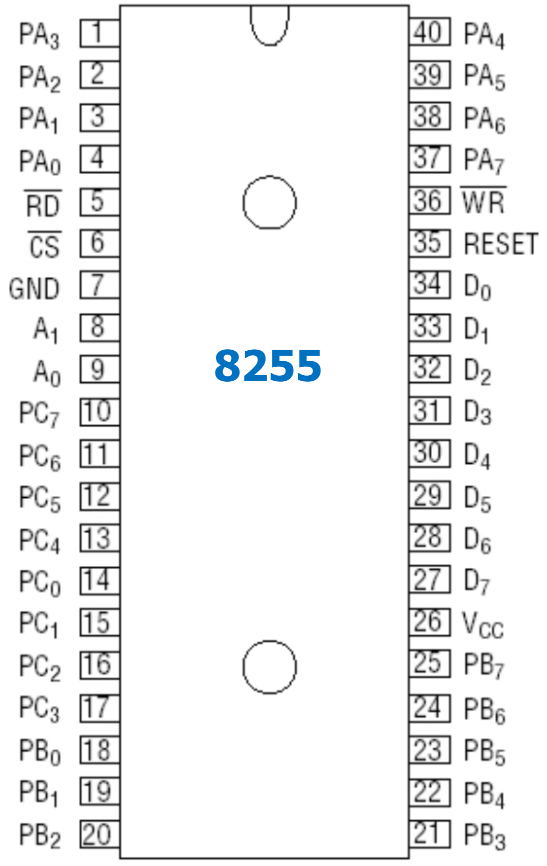
Gökhan İnce / Ayşe Yılmazer CRN: 11446

İTÜ Bilgisayar ve Bilişim Fakültesi

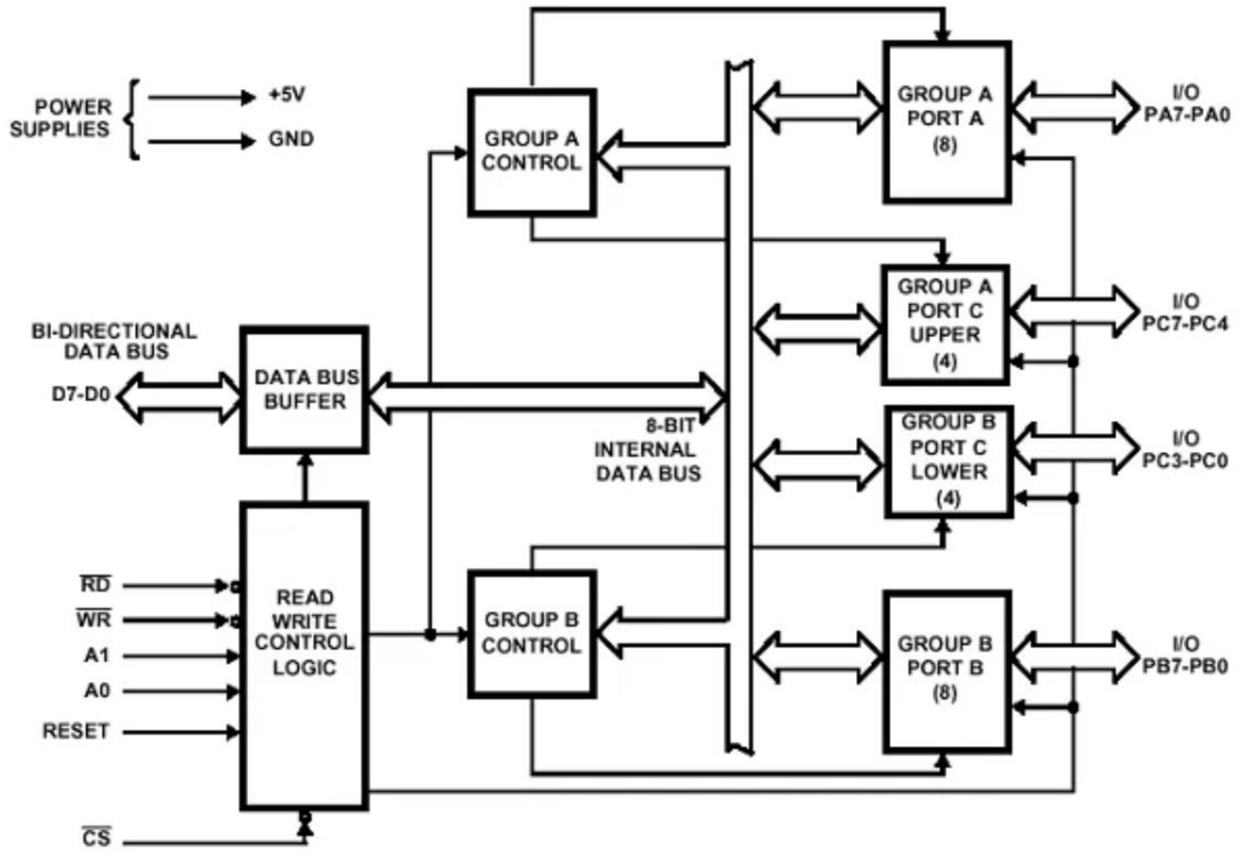
Week 15: Configurable I/O and an introduction to advanced microprocessors

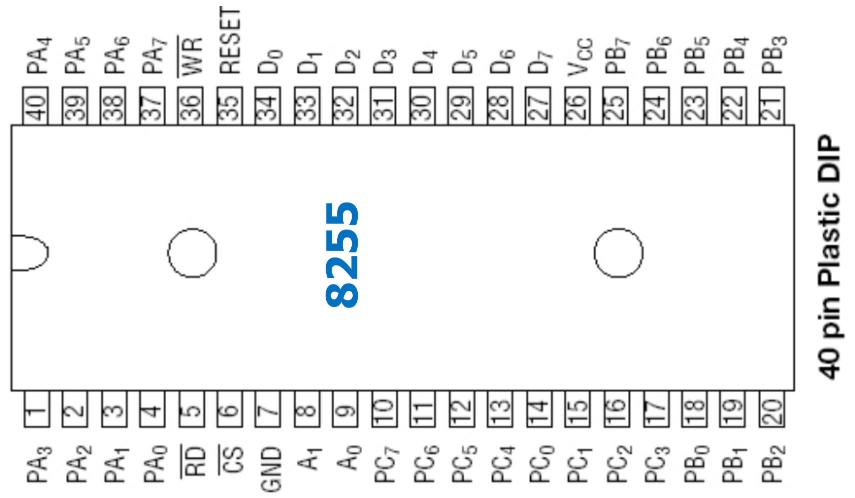
# Configurable I/O Application

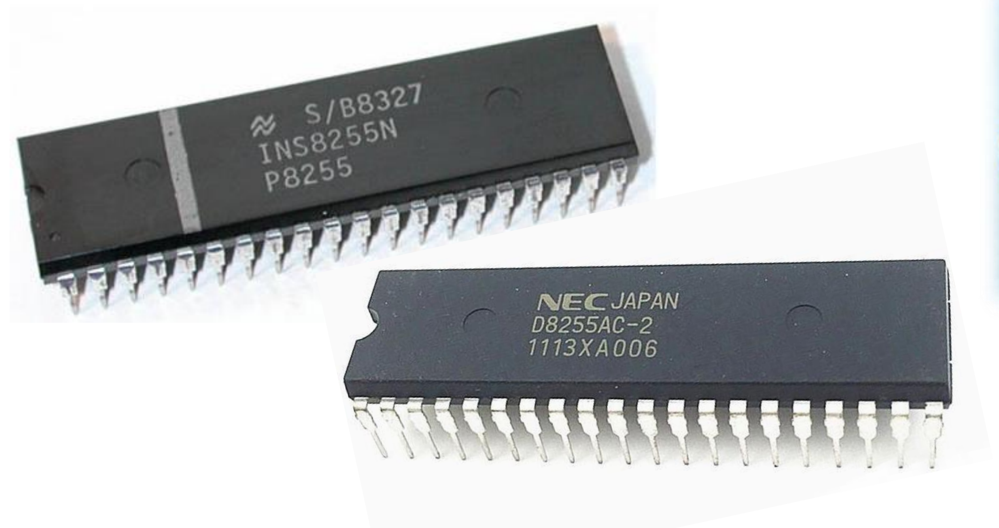
## 82C55 CMOS Programmable Peripheral Interface



40 pin Plastic DIP




8255
40 pin Plastic DIP
PA4 PA5 PA6 PA7 WR RESET D0 D1 D2 D3 D4 D5 D6 D7 VCC PB7 PB6 PB5 PB4 PB3
PA3 PA2 PA1 PA0 RD CS GND A1 A0 PC7 PC6 PC5 PC4 PC0 PC1 PC2 PC3 PB0 PB1 PB2



Russian Clone of 8255

| SYMBOL | TYPE | DESCRIPTION |
|---|---|---|
| $V_{CC}$ | | $V_{CC}$: The +5V power supply pin. A 0.1µF capacitor between $V_{CC}$ and GND is recommended for decoupling. |
| GND | | GROUND |
| D0-D7 | I/O | DATA BUS: The Data Bus lines are bidirectional three-state pins connected to the system data bus. |
| RESET | I | RESET: A high on this input clears the control register and all ports (A, B, C) are set to the input mode with the “Bus Hold” circuitry turned on. |
| $\overline{CS}$ | I | CHIP SELECT: Chip select is an active low input used to enable the 82C55A onto the Data Bus for CPU communications. |
| $\overline{RD}$ | I | READ: Read is an active low input control signal used by the CPU to read status information or data via the data bus. |
| $\overline{WR}$ | I | WRITE: Write is an active low input control signal used by the CPU to load control words and data into the 82C55A. |
| A0-A1 | I | ADDRESS: These input signals, in conjunction with the $\overline{RD}$ and $\overline{WR}$ inputs, control the selection of one of the three ports or the control word register. A0 and A1 are normally connected to the least significant bits of the Address Bus A0, A1. |
| PA0-PA7 | I/O | PORT A: 8-bit input and output port. Both bus hold high and bus hold low circuitry are present on this port. |
| PB0-PB7 | I/O | PORT B: 8-bit input and output port. Bus hold high circuitry is present on this port. |
| PC0-PC7 | I/O | PORT C: 8-bit input and output port. Bus hold circuitry is present on this port. |

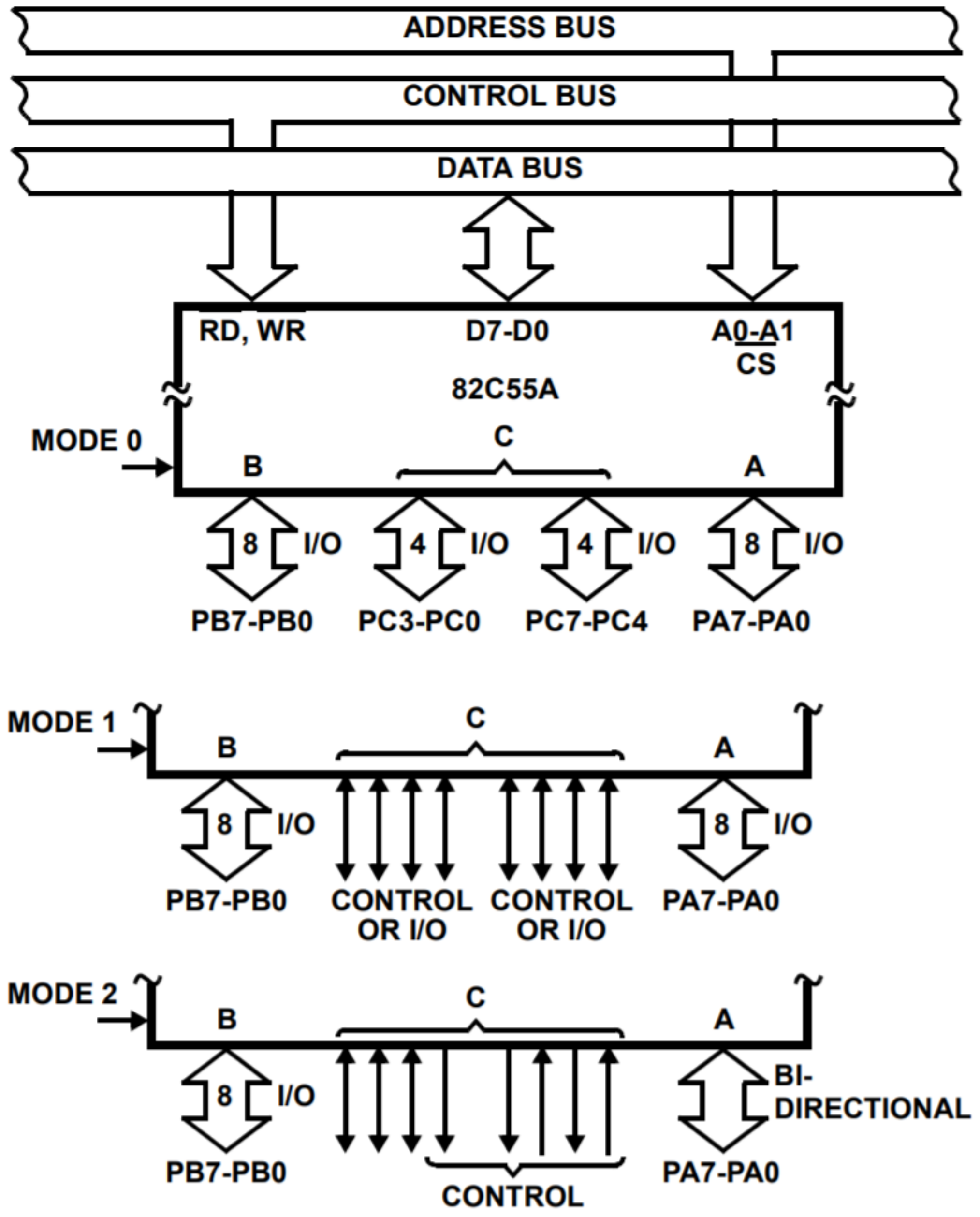
**Basic Operation:**

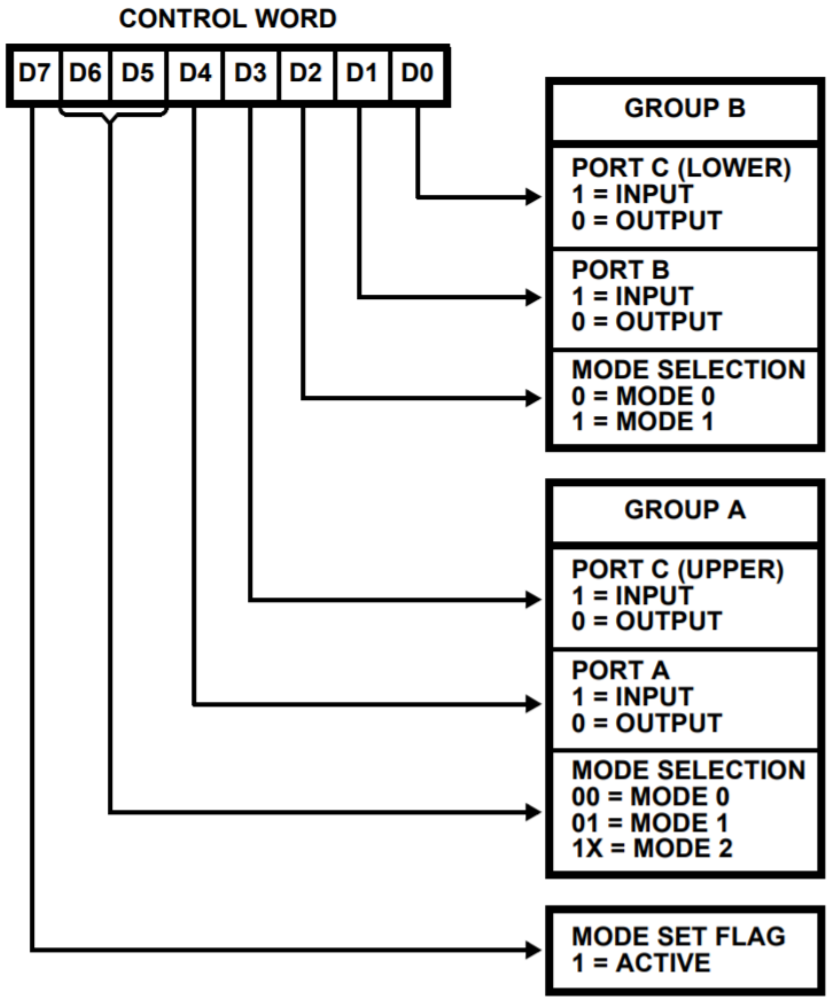
| Input Operation (READ) | | | | | |
|---|---|---|---|---|---|
| $A_1$ | $A_0$ | $\overline{RD}$ | $\overline{WR}$ | $\overline{CS}$ | |
| 0 | 0 | 0 | 1 | 0 | PORT A $\rightarrow$ Data Bus |
| 0 | 1 | 0 | 1 | 0 | PORT B $\rightarrow$ Data Bus |
| 1 | 0 | 0 | 1 | 0 | PORT C $\rightarrow$ Data Bus |

| Output Operation (WRITE) | | | | | |
|---|---|---|---|---|---|
| $A_1$ | $A_0$ | $\overline{RD}$ | $\overline{WR}$ | $\overline{CS}$ | |
| 0 | 0 | 1 | 0 | 0 | Data Bus $\rightarrow$ PORT A |
| 0 | 1 | 1 | 0 | 0 | Data Bus $\rightarrow$ PORT B |
| 1 | 0 | 1 | 0 | 0 | Data Bus $\rightarrow$ PORT C |
| 1 | 1 | 1 | 0 | 0 | Data Bus $\rightarrow$ CONTROL |

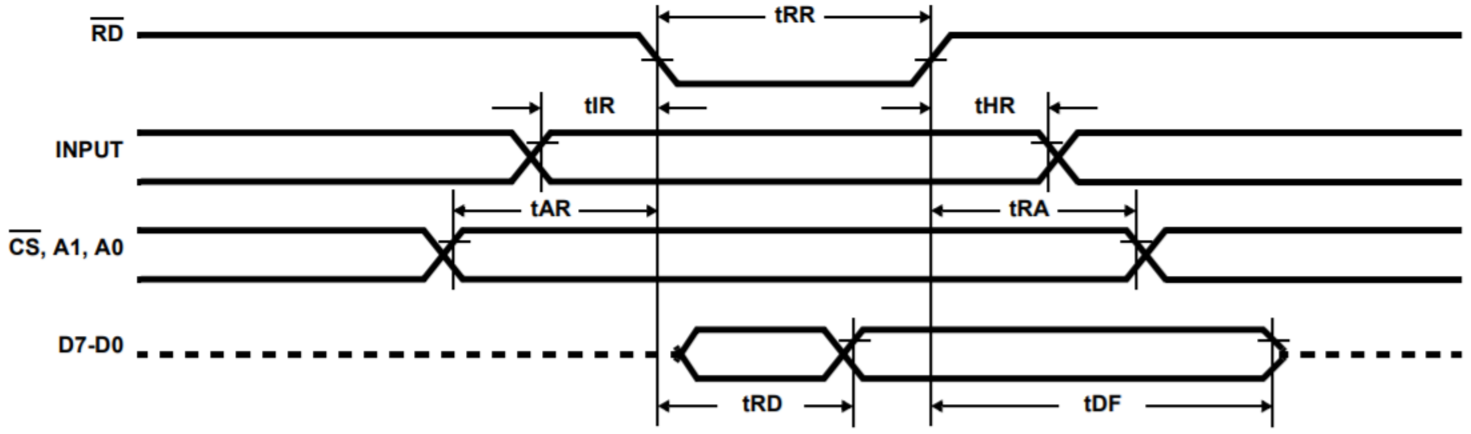
| Disable Function | | | | | |
|---|---|---|---|---|---|
| $A_1$ | $A_0$ | $\overline{RD}$ | $\overline{WR}$ | $\overline{CS}$ | |
| X | X | X | X | 1 | Data Bus $\rightarrow$ High Z State |
| X | X | 1 | 1 | 0 | Data Bus $\rightarrow$ High Z State |

## 82C55 Mode Control:

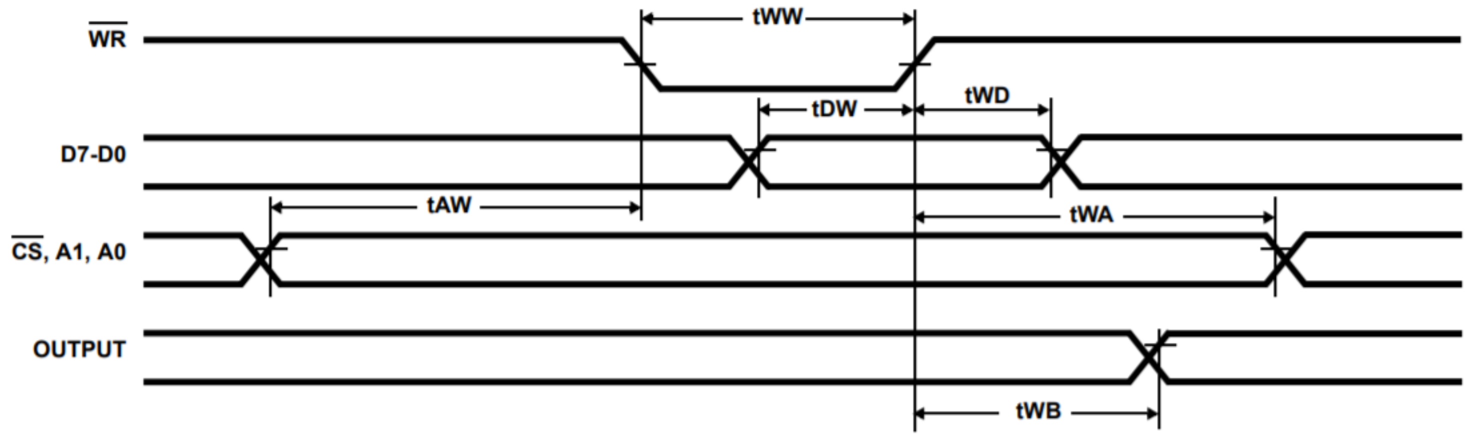




*Mode 0 (Basic Input)*



*Mode 0 (Basic Output)*

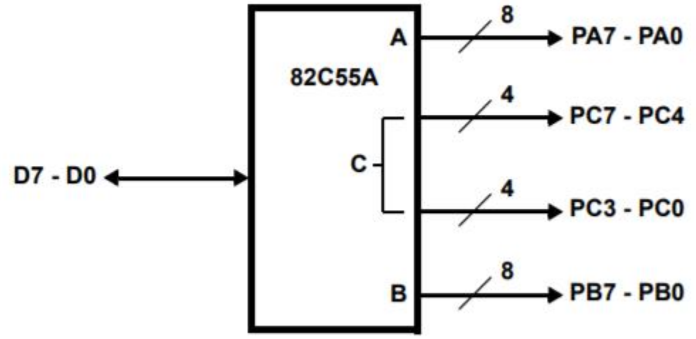


Control World Examples for 82C55

## *Mode 0 Configurations*

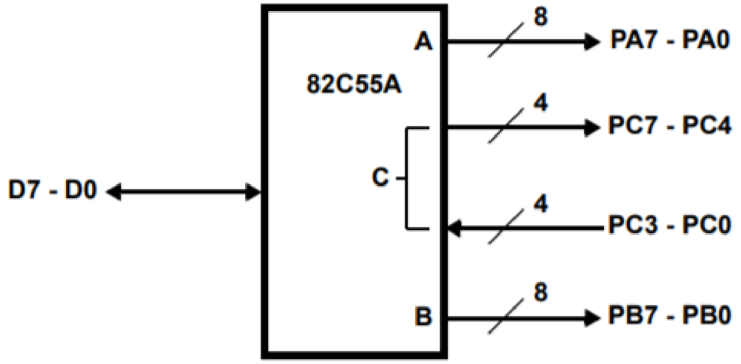
### CONTROL WORD #0

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



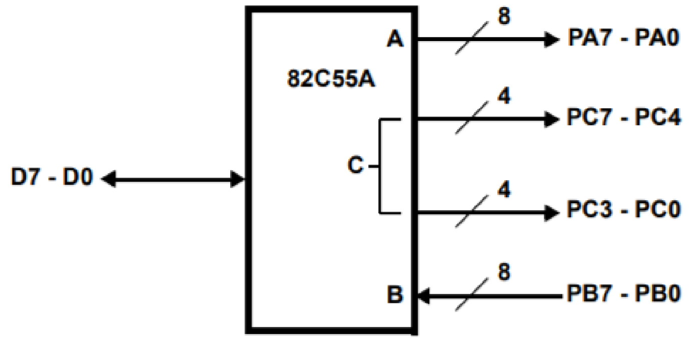
### CONTROL WORD #1

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |



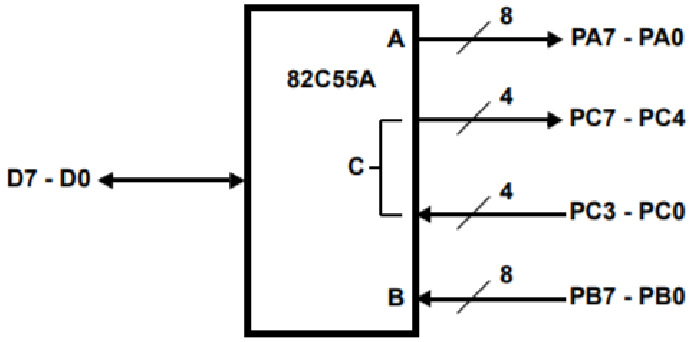
### CONTROL WORD #2

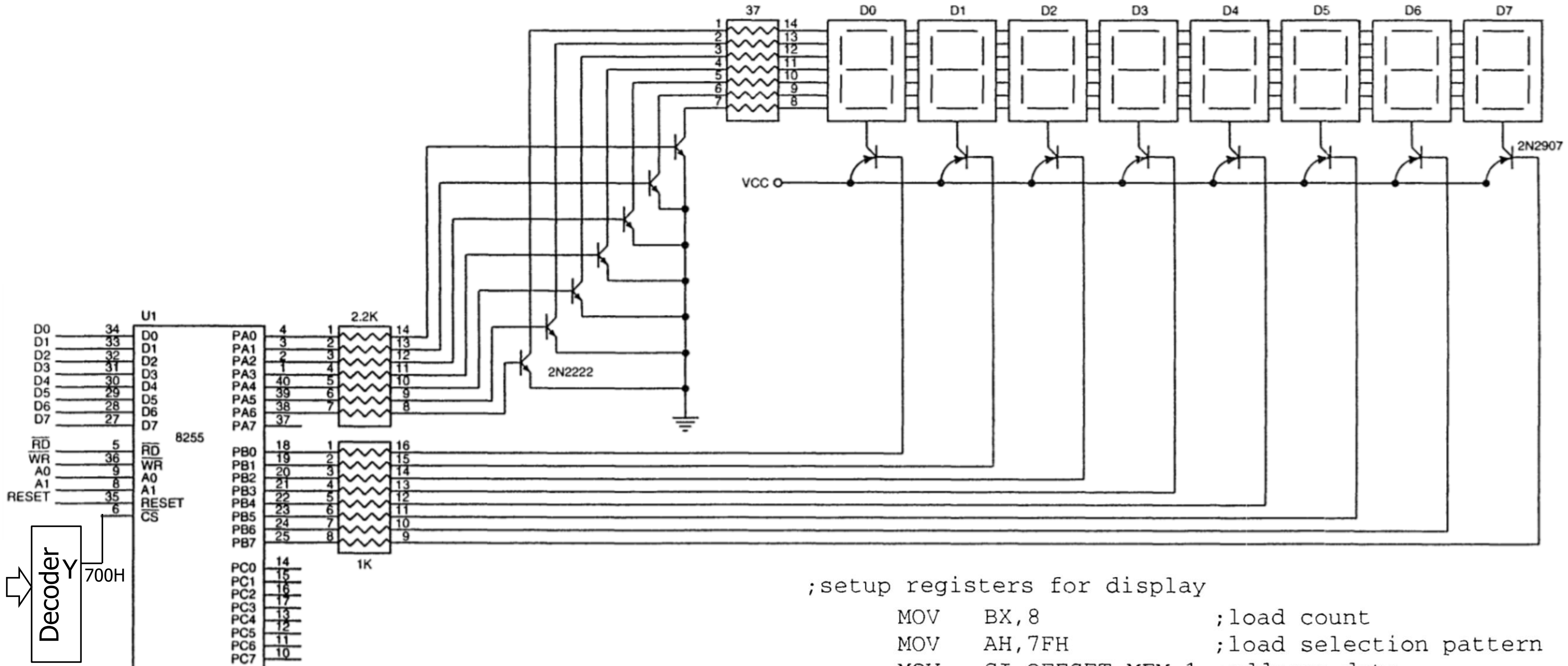
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |



### CONTROL WORD #3

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

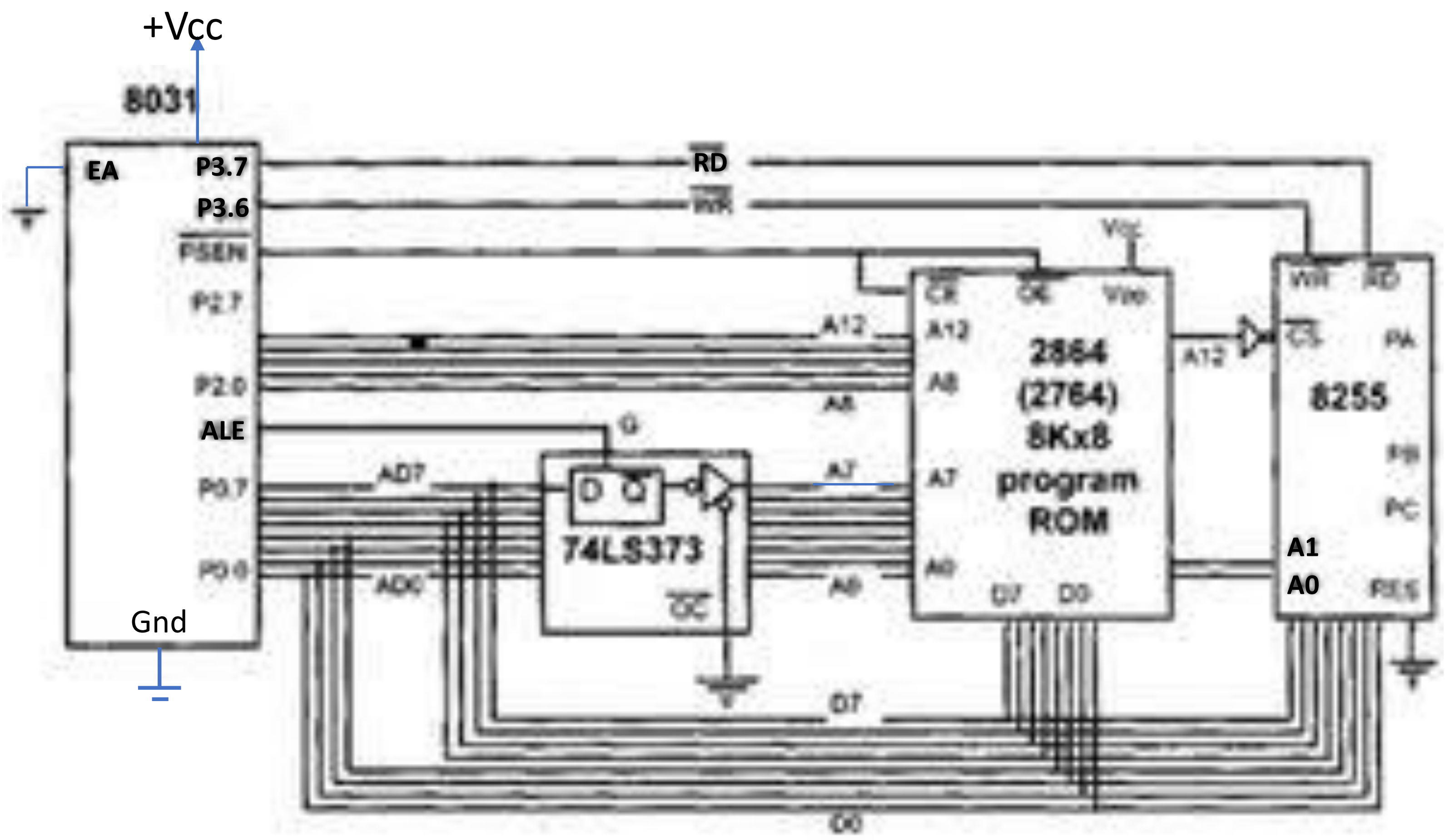




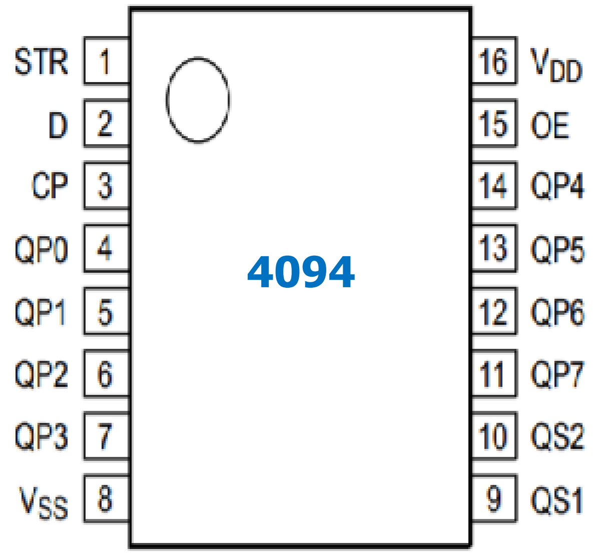
Mode 0, output configuration (INTEL):

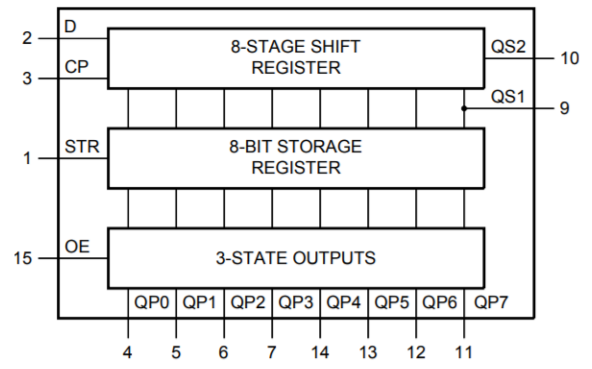
```
MOV   AL,10000000B
MOV   DX,703H            ;address command
OUT   DX,AL              ;program 82C55
```

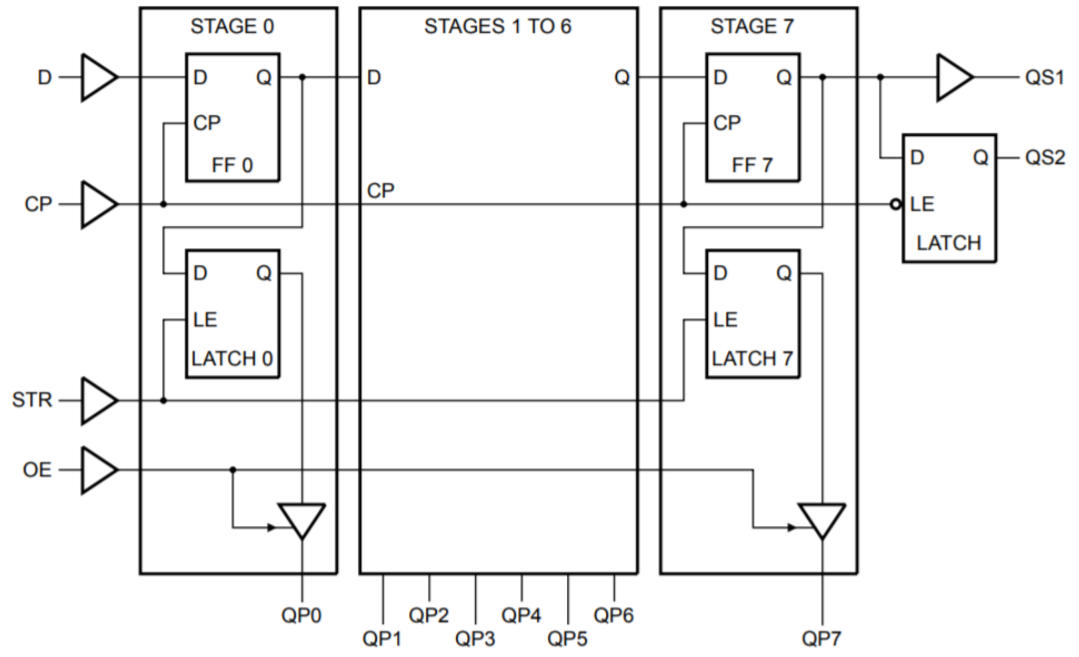
```
;setup registers for display
       MOV    BX,8              ;load count
       MOV    AH,7FH            ;load selection pattern
       MOV    SI,OFFSET MEM-1   ;address data
       MOV    DX,701H           ;address Port B
;display 8 digits
DISP1:
       MOV    AL,AH             ;select a digit
       OUT    DX,AL
       DEC    DX                ;address Port A
       MOV    AL,[BX+SI]        ;get 7-segment data
       OUT    DX,AL
       CALL   DELAY             ;wait one millisecond
       ROR    AH,1              ;address next digit
       INC    DX                ;address Port B
       DEC    BX                ;adjust count
       JNZ    DISP1             ;repeat 8 times
```


+Vcc
8031
EA
P3.7
P3.6
PSEN
P2.7
P2.0
ALE
P0.7
P0.0
Gnd
RD
WR
A12
A8
G
AD7
AD0
D Q
74LS373
OC
A7
A0
Vcc
CE
OE
Vpp
A12
A8
A7
A0
2864
(2764)
8Kx8
program
ROM
D7
D0
A12
WR
RD
CS
PA
8255
PB
PC
A1
A0
RES
D7
D0

# Serial Drive with 4094 Shitt Register

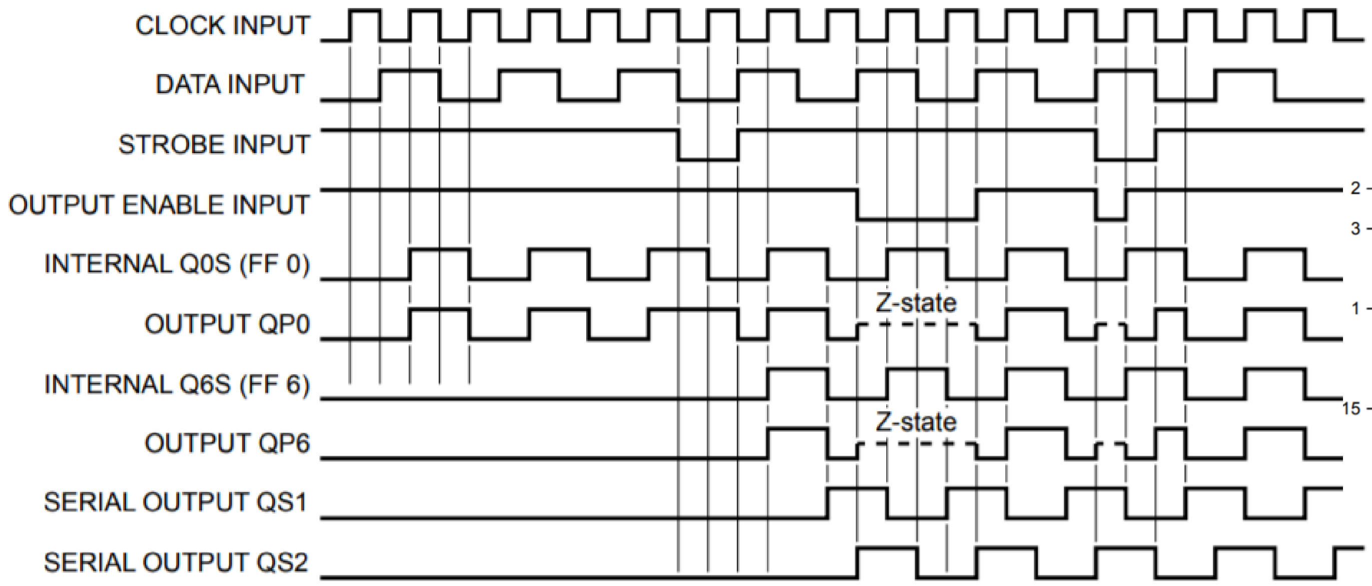


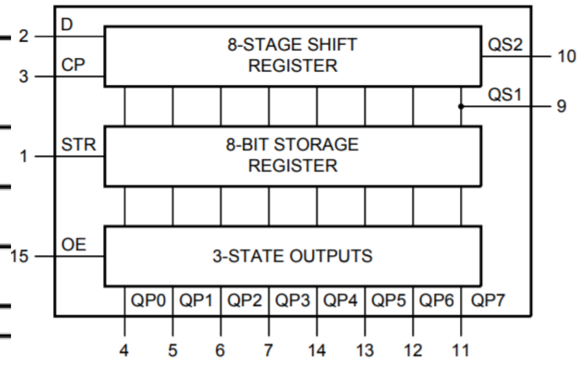


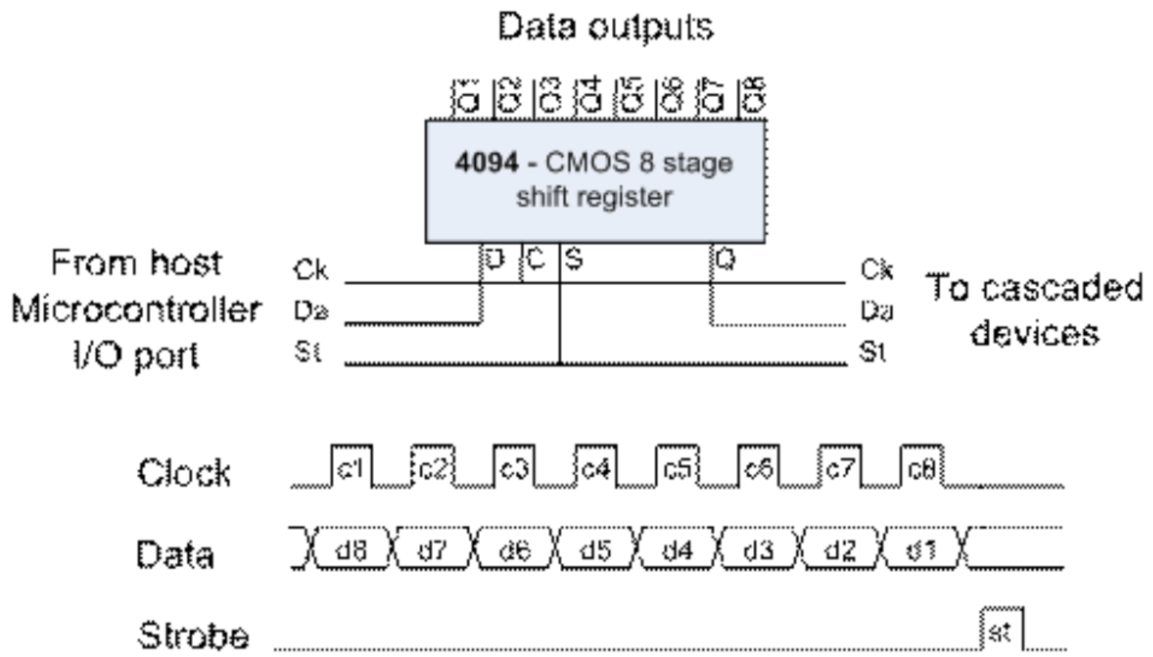


*H = HIGH voltage level; L = LOW voltage level; X = don't care; Z = HIGH-impedance OFF-state; NC = no change;*
*↑ = positive-going transition; ↓ = negative-going transition;*
*Q6S = the data in register stage 6 before the LOW to HIGH clock transition;*
*Q7S = the data in register stage 7 before the HIGH to LOW clock transition.*

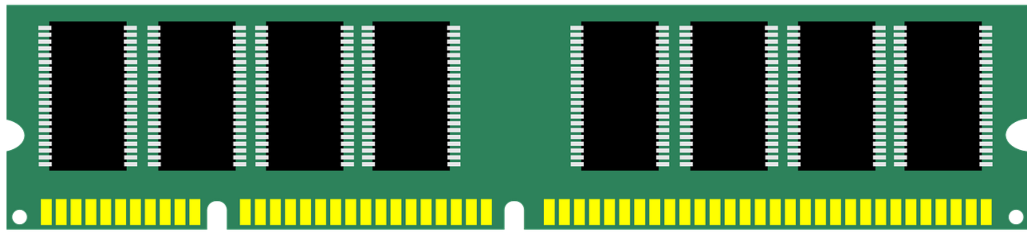
| **Inputs** | | | | **Parallel outputs** | | **Serial outputs** | |
|---|---|---|---|---|---|---|---|
| **CP** | **OE** | **STR** | **D** | **QP0** | **QPn** | **QS1** | **QS2** |
| ↑ | L | X | X | Z | Z | Q6S | NC |
| ↓ | L | X | X | Z | Z | NC | Q7S |
| ↑ | H | L | X | NC | NC | Q6S | NC |
| ↑ | H | H | L | L | QPn -1 | Q6S | NC |
| ↑ | H | H | H | H | QPn -1 | Q6S | NC |
| ↓ | H | H | H | NC | NC | NC | Q7S |






Data outputs
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
4094 - CMOS 8 stage shift register
D
C
S
Q
From host Microcontroller I/O port
Ck
Da
St
Ck
Da
St
To cascaded devices
Clock
c1
c2
c3
c4
c5
c6
c7
c8
Data
d8
d7
d6
d5
d4
d3
d2
d1
Strobe
st

# Memory Hierarchy in Modern Computers

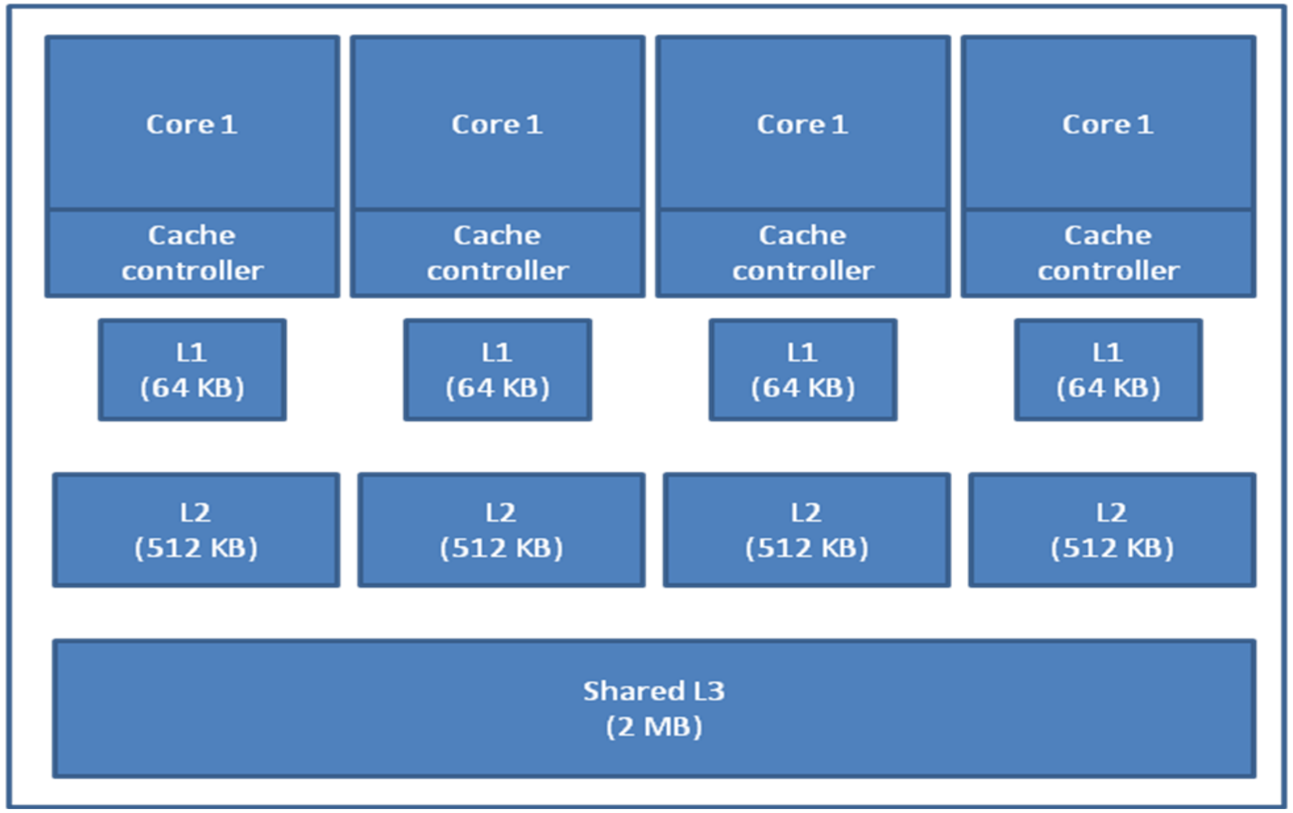


- Processors have cycle times of ~1 ns
- Fast DRAM has a cycle time of ~100 ns

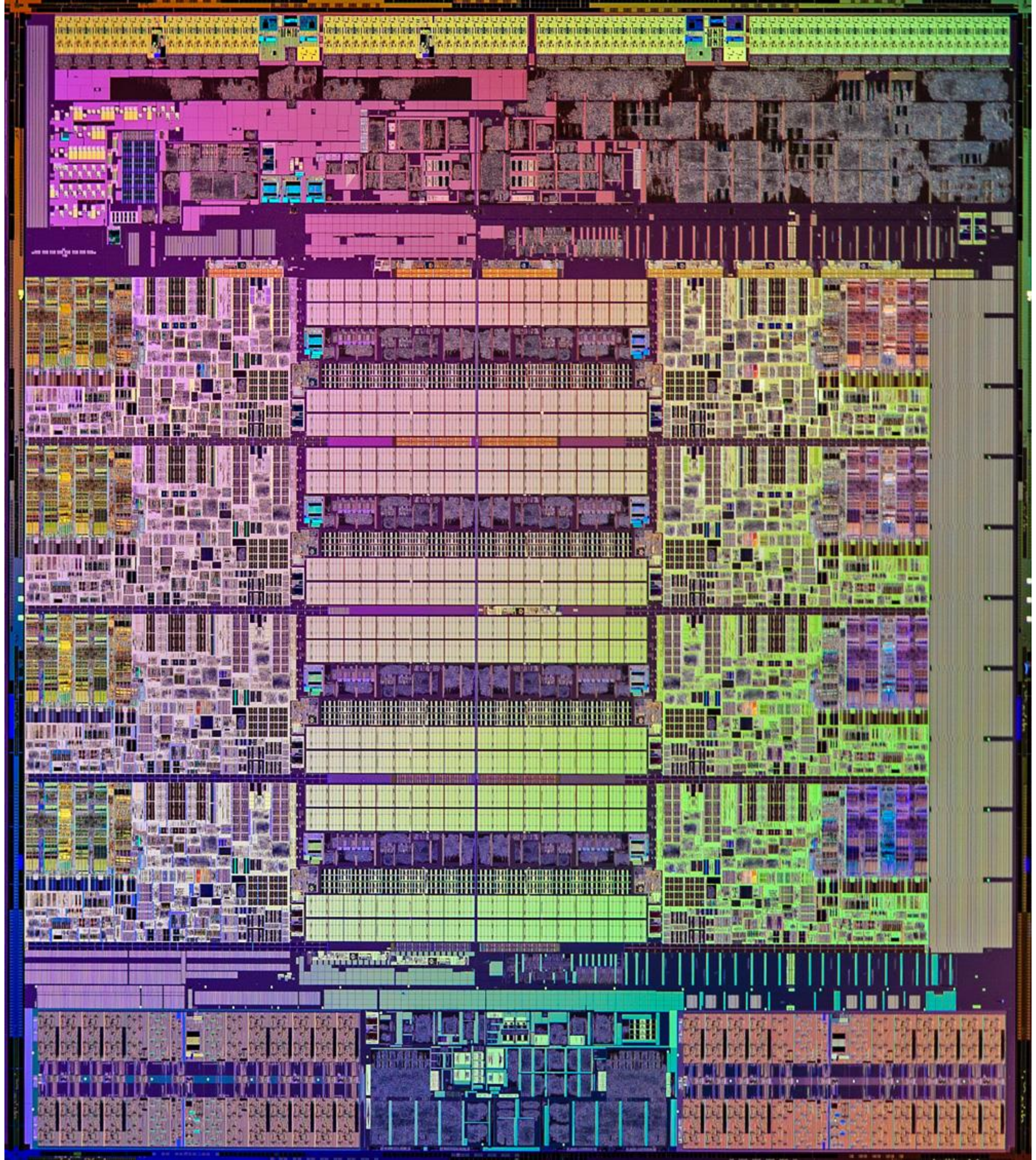
Tradeoff:

SRAM is fast

Economically unfeasible for large memories



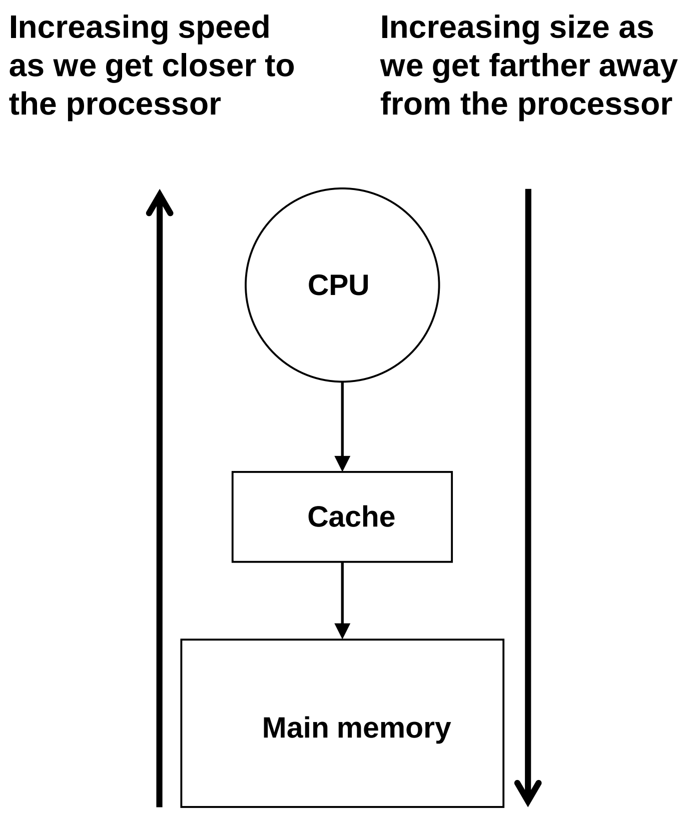
Haswell-E die shot The repetitive structures in the middle of the chip are 20MB of shared L3 cache.



## The Concept of a Cache

- Feasible to have small amount of fast memory and/or large amount of slow memory.
- Target:
  - Size advantage of DRAM
  - Speed advantage of SRAM.

- CPU looks in cache for data it seeks from main memory.
- If data not there it retrieves it from main memory.
- If the cache is able to service "most" CPU requests then effectively we will get speed advantage of cache.
- All addresses in cache are also in memory


Increasing speed as we get closer to the processor
Increasing size as we get farther away from the processor
CPU
Cache
Main memory

# Principle of Locality

- A program tends to access a relatively small region of memory irrespective of its actual memory footprint in any given interval of time. While the region of activity may change over time, such changes are gradual
- Spatial Locality: Tendency for locations close to a location that has been accessed to also be accessed
- Temporal Locality: Tendency for a location that has been accessed to be accessed again
- Example

```
    for(i=0; i<100000; i++)
        a[i] = b[i];
```

## Basic terminologies

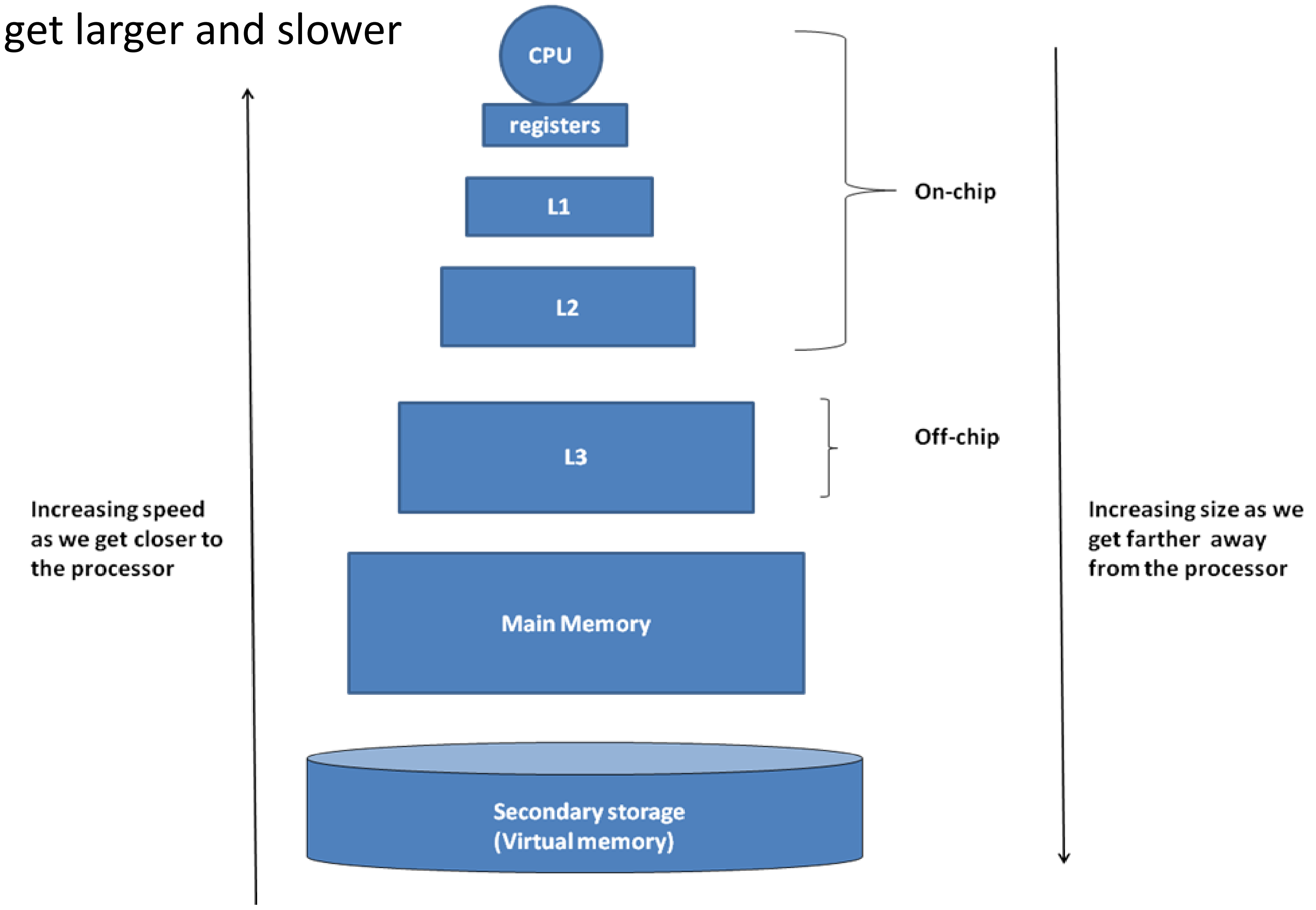
- ***Hit***: CPU finding contents of memory address in cache
- ***Hit rate*** *(h)* is probability of *successful lookup* in cache by CPU.
- ***Miss***: CPU *failing* to find what it wants in cache (incurs trip to deeper levels of memory hierarchy
- ***Miss rate*** *(m)* is probability of *missing* in cache and is equal to *1-h*.
- ***Miss penalty***: Time penalty associated with servicing a miss at any particular level of memory hierarchy
- ***Effective Memory Access Time (EMAT)***: Effective access time experienced by the CPU when accessing memory.
  - Time to lookup cache to see if memory location is already there
  - Upon cache miss, time to go to deeper levels of memory hierarchy

*EMAT = Tc + m * Tm*

*where m is cache miss rate, Tc* the cache access time and *Tm* the miss penalty

# Multilevel Memory Hierarchy

- Modern processors use multiple levels of caches.
- As we move away from processor, caches get larger and slower
- $\text{EMAT}_i = T_i + m_i * \text{EMAT}_{i+1}$
- where $T_i$ is access time for level i
- and $m_i$ is miss rate for level i


CPU
registers
L1
L2
L3
Main Memory
Secondary storage (Virtual memory)
On-chip
Off-chip
Increasing speed as we get closer to the processor
Increasing size as we get farther away from the processor

## Cache organization

- There are three facets to the organization of the cache:
  1. Placement: Where do we place in the cache the data read from the memory?
  2. Algorithm for lookup: How do we find something that we have placed in the cache?
  3. Validity: How do we know if the data in the cache is valid?

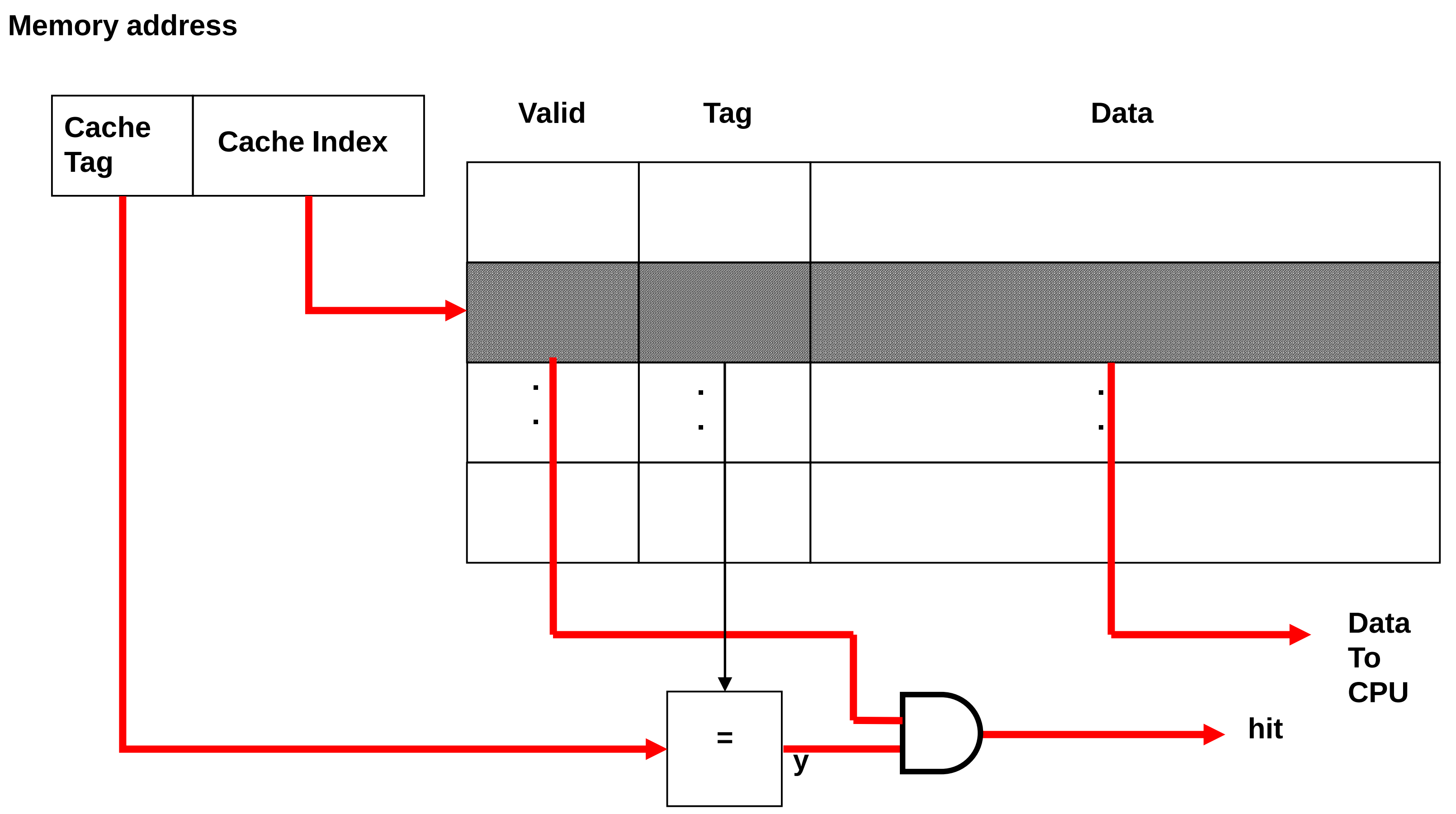
## Cache Lookup

- Example:
  - 4Gb Memory: 32 bit address
  - 256 Kb Cache
  - Cache is organized by words
    - 1 Gword memory
    - 64 Kword cache ➔ 16  bit cache index

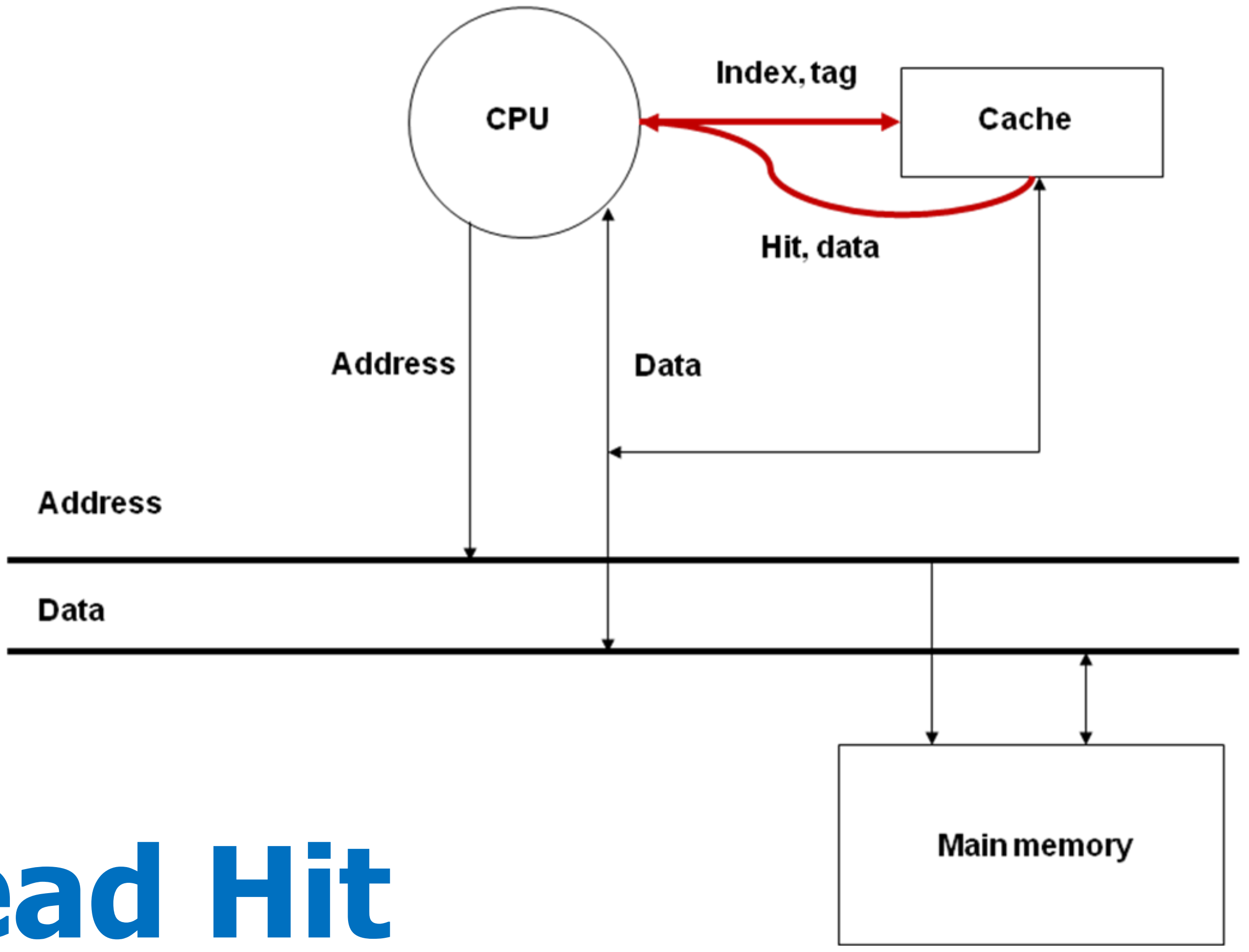
| **Memory Address Breakdown** | Tag<br>00000000000000 | Index<br>0000000000000000 | Byte Offset<br>00 |
|---|---|---|---|

| **Cache Line** | Index<br>0000000000000000 | Tag<br>00000000000000 | Contents<br>00000000000000000000000000000000 |
|---|---|---|---|

# Hardware for direct mapped cache



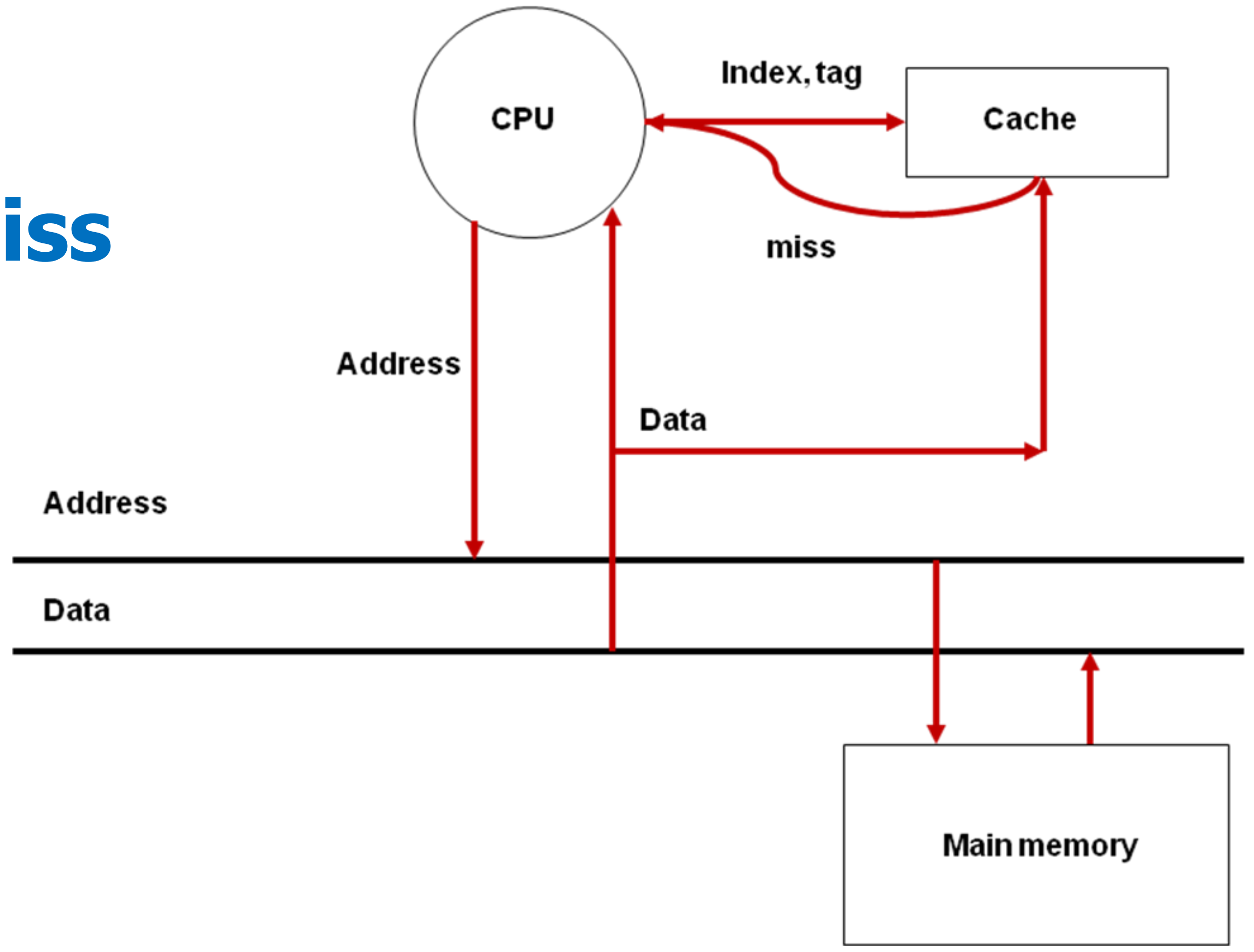
# Cache read/write algorithms



**Read Hit**

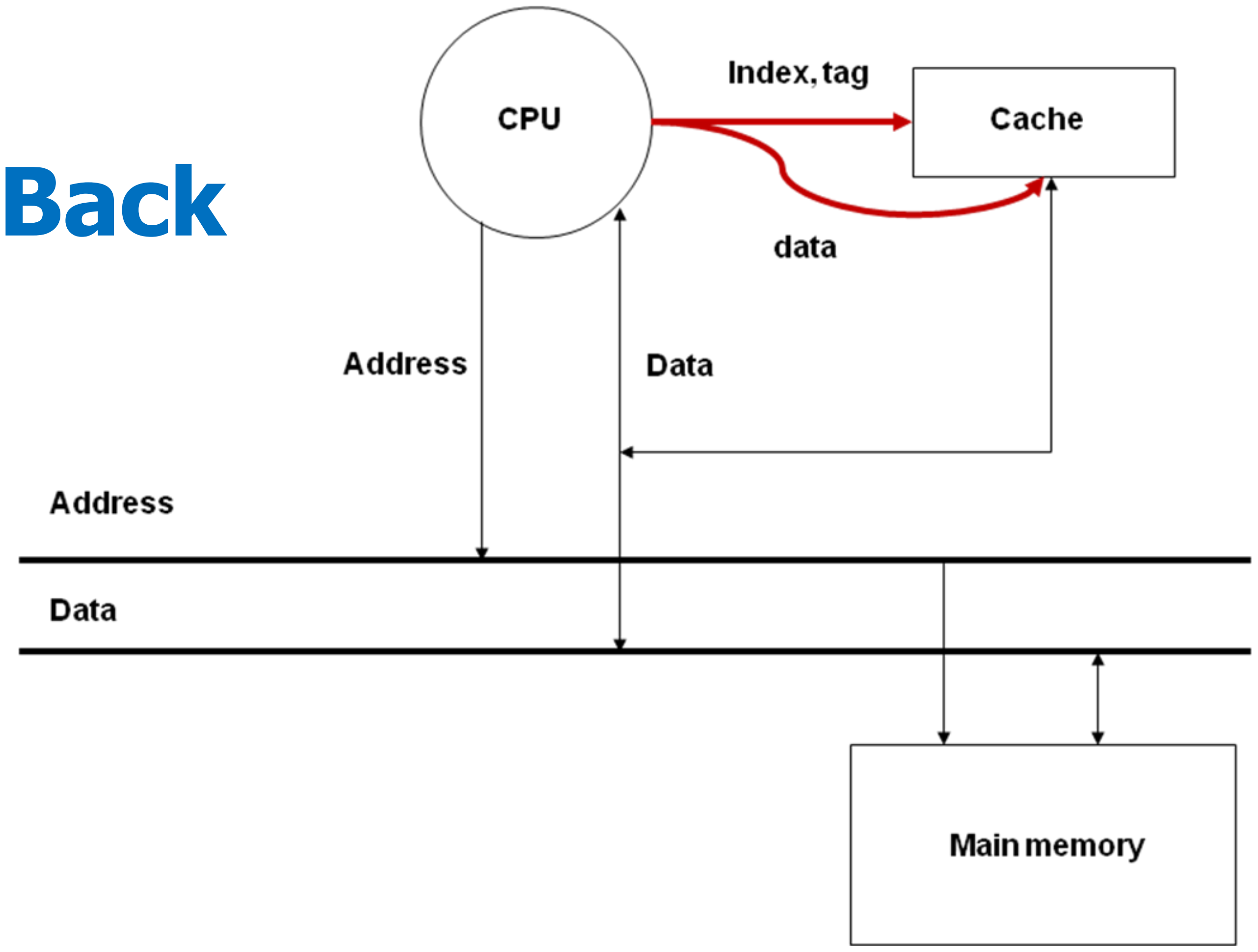
## Basic cache read/write algorithms

**Read Miss**



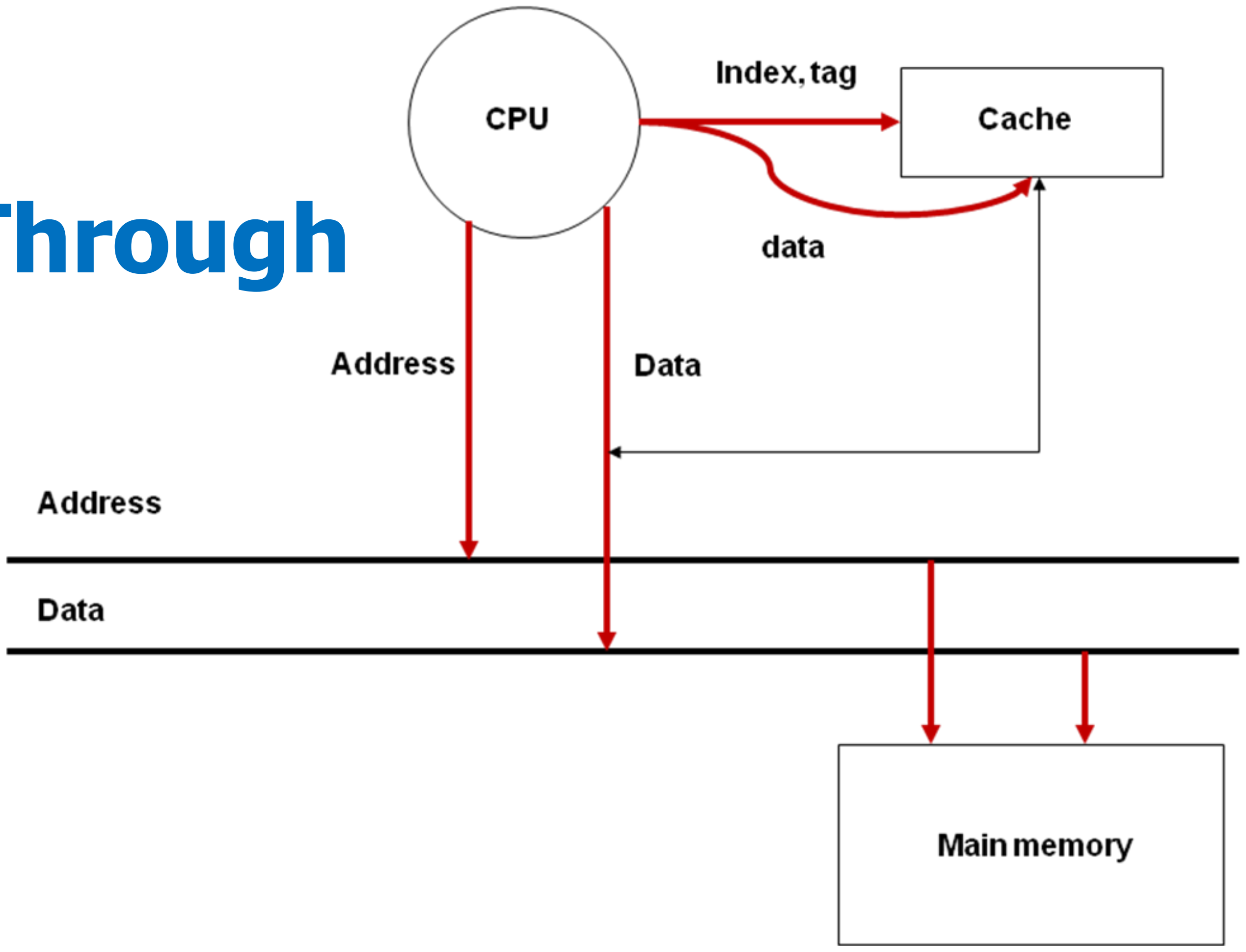
## Basic cache read/write algorithms

**Write-Back**



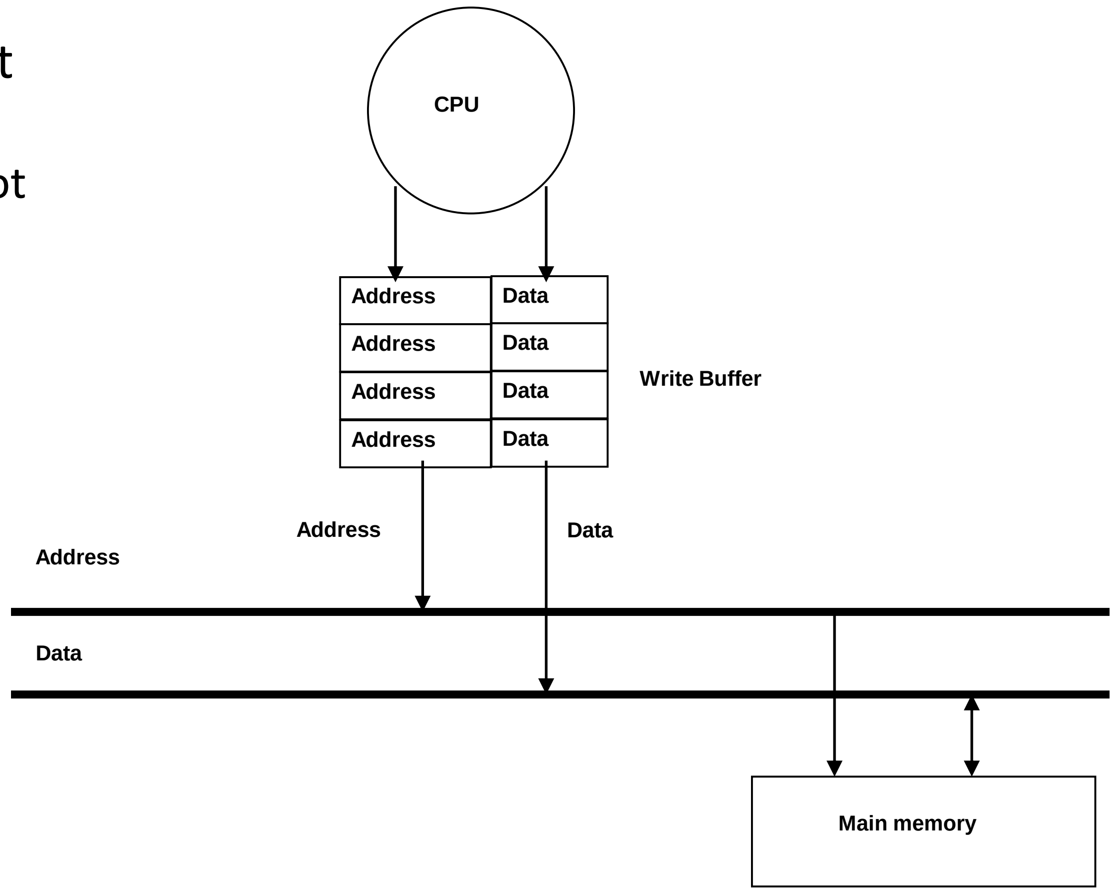
# Basic cache read/write algorithms

**Write-Through**



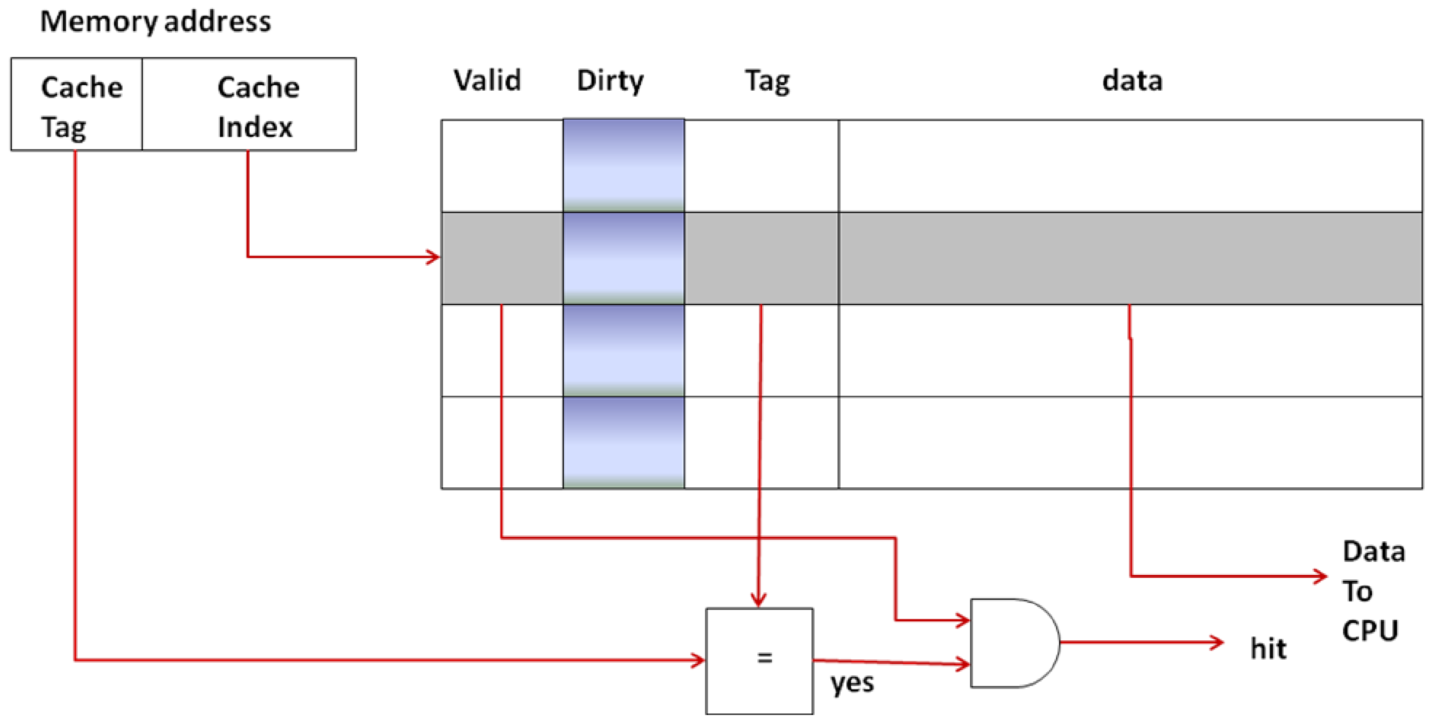
## Write Through Policy

- Each write goes to cache. Tag is set and valid bit is set
    - This is write allocate
    - There is also a no-write allocate where the cache is not written to if there was a write miss
- Each write also goes to write buffer
- Write buffer writes data into main memory
    - Will stall if write buffer full



# Write back policy

- CPU writes data to cache setting dirty bit
  - Note: Cache and memory are now inconsistent but the dirty bit tells us that



- Write Through
  - Cache logic simpler and faster
  - Creates more bus traffic
- Write back
  - Requires dirty bit and extra logic
- Multilevel cache processors may use both
  - L1 Write through
  - L2/L3 Write back

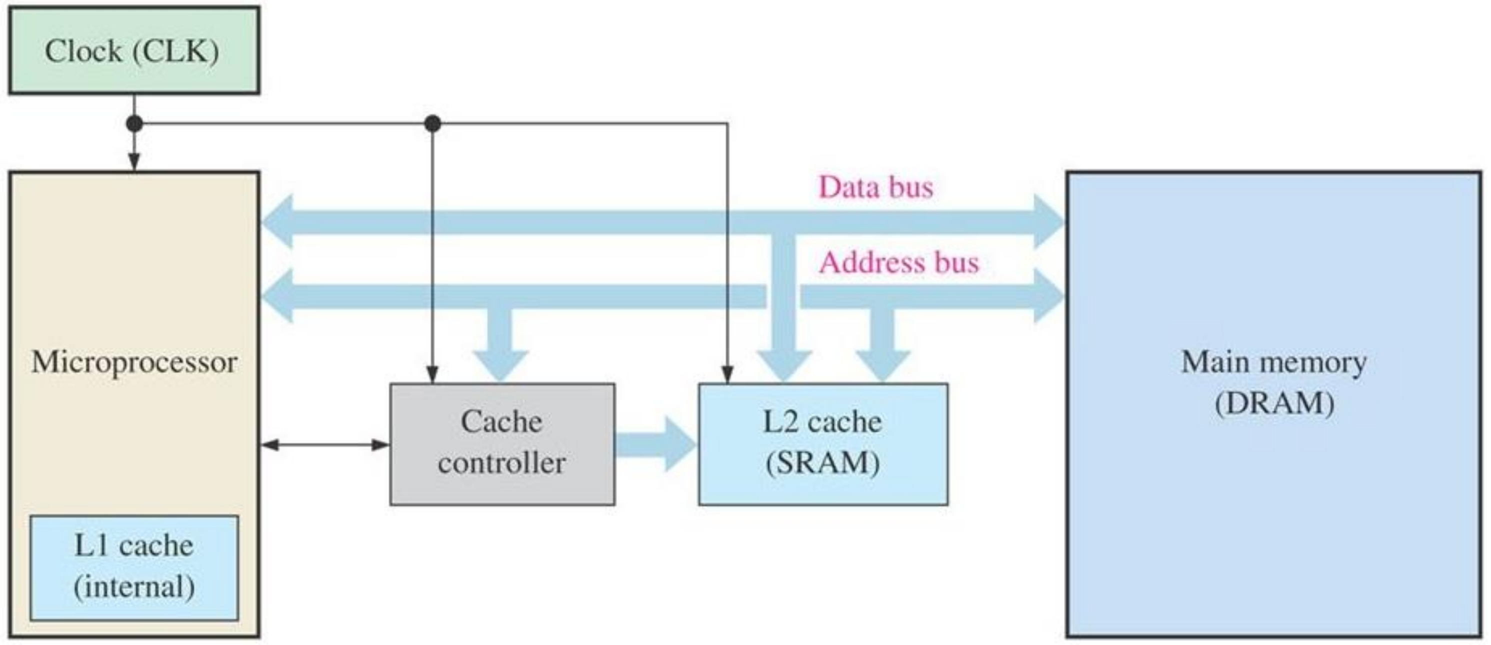
## Cache misses in the processor pipeline

- **Read miss in the MEM stage:**

```
I1: ld  r1, a          ; r1 <- MEM[a]
I2: add r3, r4, r5     ;   r3 <- r4 + r5
I3: and r6, r7, r8     ;   r6 <- r7 AND r8
I4: add r2, r4, r5     ;   r2 <- r4 + r5
I5: add r2, r1, r2     ;   r2 <- r1 + r2
```

- **Write miss in the MEM stage:** The write-buffer alleviates the ill effects of write misses in the MEM stage. (Write-Through)

# Cache controller



- Upon request from processor, looks up cache to determine hit or miss, serving data up to processor in case of hit.
- Upon miss, initiates bus transaction to read missing block from deeper levels of memory hierarchy.
- Depending on details of memory bus, requested data block may arrive asynchronously with respect to request.  In this case, cache controller receives block and places it in appropriate spot in cache.
- Provides ability for the processor to specify certain regions of memory as “uncachable.”

## Cache Design Considerations

- Principles of spatial and temporal locality
- Hit, miss, hit rate, miss rate, cycle time, hit time, miss penalty
- Multilevel caches and design considerations thereof
- Direct mapped caches
- Cache read/write algorithms
- Spatial locality and blocksize
- Fully- and set-associative caches
- Considerations for I- and D-caches
- Cache replacement policy
- Types of misses
- TLB and caches
- Cache controller
- Virtually indexed physically tagged caches

| Category | Vocabulary | Details |
|---|---|---|
| **Principle of locality** | Spatial | Access to contiguous memory locations |
| | Temporal | Reuse of memory locations already accessed |
| **Cache organization** | Direct-mapped | One-to-one mapping |
| | Fully associative | One-to-any mapping |
| | Set associative | One-to-many mapping |
| **Cache reading/writing** | Read hit/Write hit | Memory location being accessed by the CPU is present in the cache |
| | Read miss/Write miss | Memory location being accessed by the CPU is not present in the cache |
| **Cache write policy** | Write through | CPU writes to cache and memory |
| | Write back | CPU only writes to cache; memory updated on replacement |
| **Cache parameters** | Total cache size ($S$) | Total data size of cache in bytes |
| | Block Size ($B$) | Size of contiguous data in one data block |
| | Degree of associativity ($p$) | Number of homes a given memory block can reside in a cache |
| | Number of cache lines ($L$) | $S/pB$ |
| | Cache access time | Time in CPU clock cycles to check hit/miss in cache |
| | Unit of CPU access | Size of data exchange between CPU and cache |
| | Unit of memory transfer | Size of data exchange between cache and memory |
| | Miss penalty | Time in CPU clock cycles to handle a cache miss |
| **Memory address interpretation** | Index ($n$) | $log_2 L$ bits, used to look up a particular cache line |
| | Block offset ($b$) | $log_2 B$ bits, used to select a specific byte within a block |
| | Tag ($t$) | $a – (n+b)$ bits, where $a$ is number of bits in memory address; used for matching with tag stored in the cache |

## An example memory hierarchy of modern microprocessors

| **Type of Memory** | **Typical Size** | **Approximate latency in CPU clock cycles to read one word of 4 bytes** |
|---|---|---|
| CPU registers | 8 to 32 | Usually immediate access (0-1 clock cycles) |
| L1 Cache | 32 (Kilobyte) KB to 128 KB | 3 clock cycles |
| L2 Cache | 128 KB to 4 Megabyte (MB) | 10 clock cycles |
| Main (Physical) Memory | 256 MB to 4 Gigabyte (GB) | 100 clock cycles |
| Virtual Memory (on disk) | 1 GB to 1 Terabyte (TB) | 1000 to 10,000 clock cycles (not accounting for the software overhead of handling page faults) |

| Cache | Bulldozer | Piledriver | Steamroller |
|---|---|---|---|
| Level 1 code | 64 kB, 2-way, 64 B line size, shared between two cores. | 64 kB, 2-way, 64 B line size, shared between two cores. | 96 kB, 3-way, 64 B line size, shared between two cores. |
| Level 1 data | 16 kB, 4-way, 64 B line size, per core. Latency 3-4 clocks. | 16 kB, 4-way, 64 B line size, per core. Latency 3-4 clocks. | 16 kB, 4-way, 64 B line size, per core. Latency 3-4 clocks. |
| Level 2 | 1 - 2 MB, 16-way, 64 B line size, shared between two cores. Latency 21 clocks. Read throughput 1 per 4 clock. Write throughput 1 per 12 clock. | 2 MB, 16-way, 64 B line size, shared between two cores. Latency 20 clocks. Read throughput 1 per 4 clock. Write throughput 1 per 12 clock. | 2 MB, 16-way, 64 B line size, shared between two cores. Latency 19 clocks. Read throughput 1 per 4 clock. Write throughput 1 per 6 clock. |
| Level 3 | 0 - 8 MB, 64-way, 64 B line size, shared between all cores. Latency 87 clock. Read throughput 1 per 15 clock. Write throughput 1 per 21 clock. | 0 - 8 MB, 64-way, 64 B line size, shared between all cores. Latency 87 clock. Read throughput 1 per 15 clock. Write throughput 1 per 21 clock. | None |

**Table Cache sizes on AMD Bulldozer, Piledriver and Steamroller**