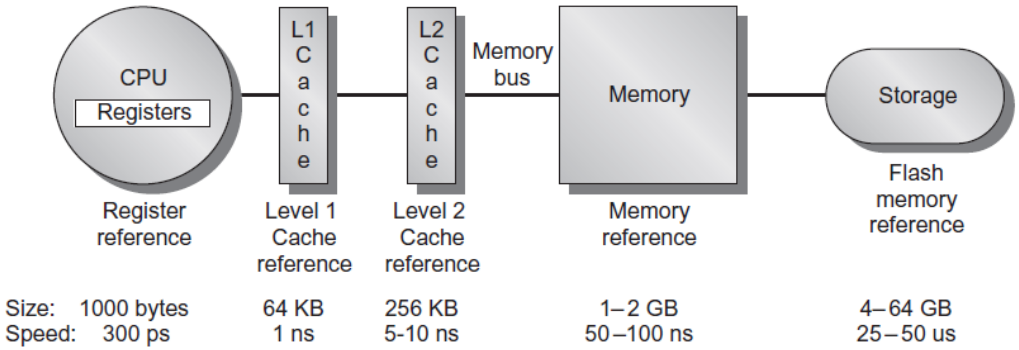
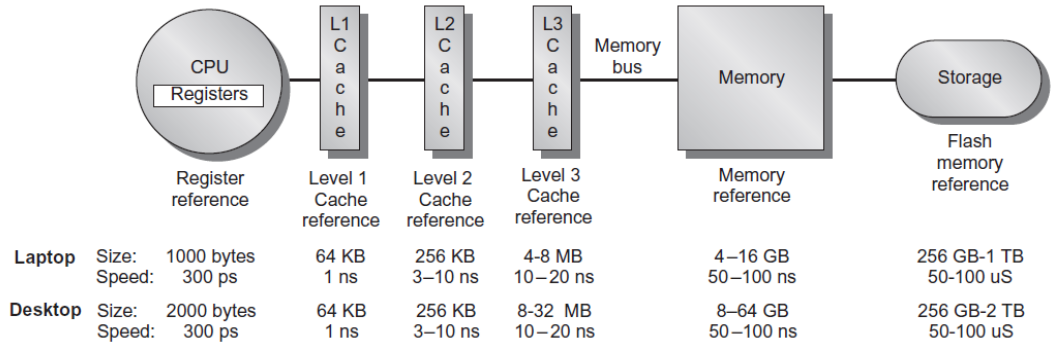
Memory Hierarchy Design

Presented by: Han Trung Dinh

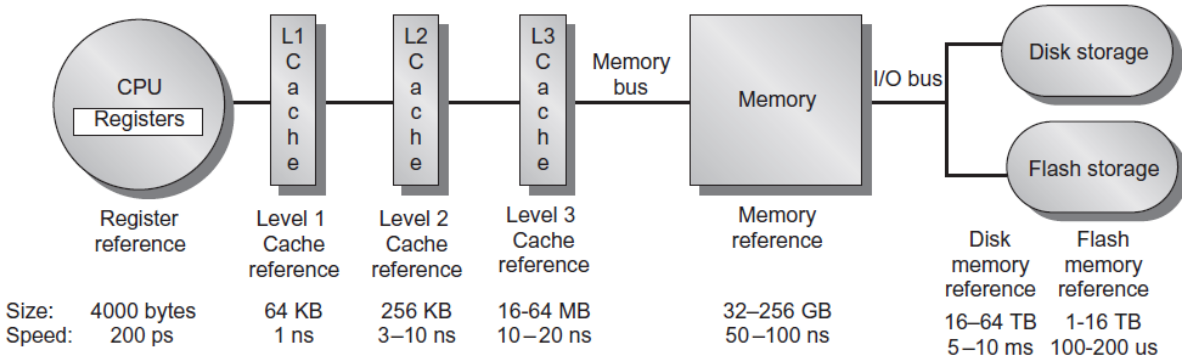
The levels in a typical memory hierarchy



(a) Memory hierarchy for a personal mobile device



(b) Memory hierarchy for a laptop or a desktop



(c) Memory hierarchy for server

Fig. 1. Typical memory hierarchy

Basics of Memory Hierarchies

- Compulsory: The very first access to a block cannot be in the cache
- Capacity: If the cache cannot contain all the blocks needed during execution of a program. Blocks are being discarded and later retrieved
- Conflict: Conflict misses will occur because a block may be discarded and later retrieved if multiple blocks map to its set and accesses to the different blocks are intermingled

$$\frac{\text{Misses}}{\text{Instruction}} = \frac{\text{Miss rate} \times \text{Memory accesses}}{\text{Instruction count}} = \text{Miss rate} \times \frac{\text{Memory accesses}}{\text{Instruction}}$$

$$\text{Average memory access time} = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$$

Six basic cache optimizations

- Larger block size to reduce miss rate
- Bigger caches to reduce miss rate
- Higher associativity to reduce miss rate
- Multilevel caches to reduce miss penalty
- Giving priority to read misses over writes to reduce miss penalty
- Avoiding address translation during indexing of the cache to reduce hit time

Memory Technology and Optimizations

- Access time is the time between when a read is requested and when the desired word arrives
- Cycle time is the minimum time between unrelated requests to memory.
- SRAM (Static Random Access Memory): don't need to refresh, so the access time is very close to the cycle time

Memory Technology and Optimizations

- DRAM (Dynamic Random Access Memory): need to refresh

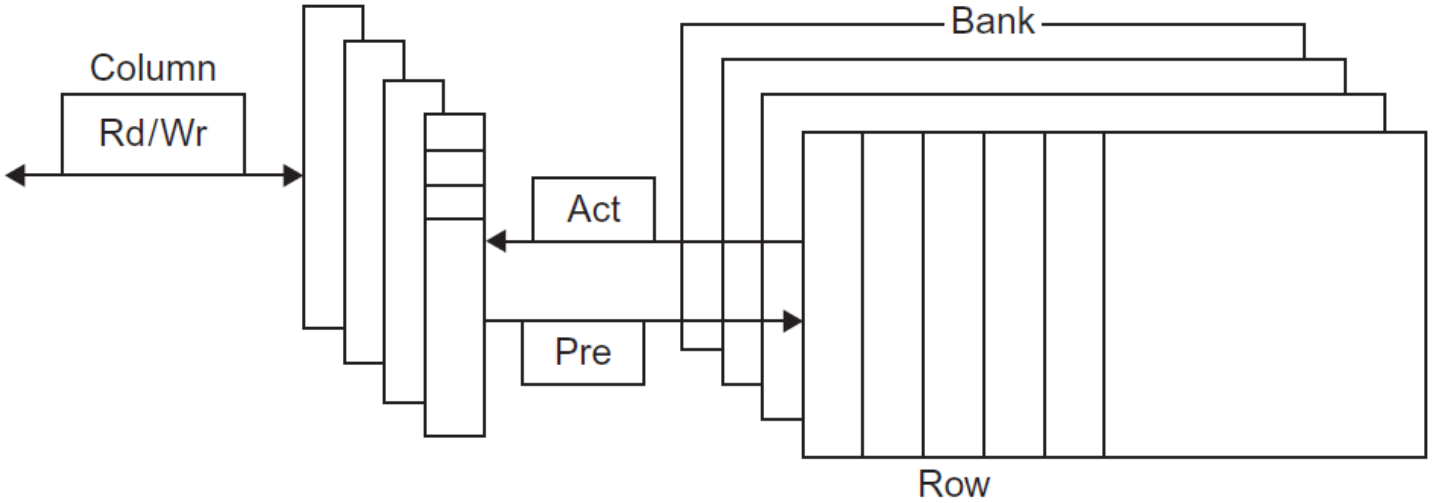


Fig. 2. Internal organization of a DRAM

Note:
- One-half of the address is sent first during the row access strobe (RAS)
- The other half of the address, sent during the column access strobe (CAS)

- DRAMs are organized in banks
- Each bank consists of a series of rows
- Sending an ACT (Activate) command opens a bank and a row and loads the row into a row buffer
- The Precharge commend (PRE) closes the bank and row and readies it for a new access
- Each command, as well as block transfers, are synchronized with a clock

Memory Technology and Optimizations

- SDRAM (synchronous DRAM): Improving memory performance inside a DRAM Chip by adding a clock signal to avoid sync with the controller overhead and offering Burst Transfer Mode

			Best case access time (no precharge)			Precharge needed
Production year	Chip size	DRAM type	RAS time (ns)	CAS time (ns)	Total (ns)	Total (ns)
2000	256M bit	DDR1	21	21	42	63
2002	512M bit	DDR1	15	15	30	45
2004	1G bit	DDR2	15	15	30	45
2006	2G bit	DDR2	10	10	20	30
2010	4G bit	DDR3	13	13	26	39
2016	8G bit	DDR4	13	13	26	39

Fig. 3. Capacity and access times for DDR SDRAMs by year of production

Memory Technology and Optimizations

- DIMM: Dual Inline Memory Module

Standard	I/O clock rate	M transfers/s	DRAM name	MiB/s/DIMM	DIMM name
DDR1	133	266	DDR266	2128	PC2100
DDR1	150	300	DDR300	2400	PC2400
DDR1	200	400	DDR400	3200	PC3200
DDR2	266	533	DDR2-533	4264	PC4300
DDR2	333	667	DDR2-667	5336	PC5300
DDR2	400	800	DDR2-800	6400	PC6400
DDR3	533	1066	DDR3-1066	8528	PC8500
DDR3	666	1333	DDR3-1333	10,664	PC10700
DDR3	800	1600	DDR3-1600	12,800	PC12800
DDR4	1333	2666	DDR4-2666	21,300	PC21300

Fig. 4. Clock rates, bandwidth, and names of DDR DRAMS and DIMMs in 2016

Memory Technology and Optimizations

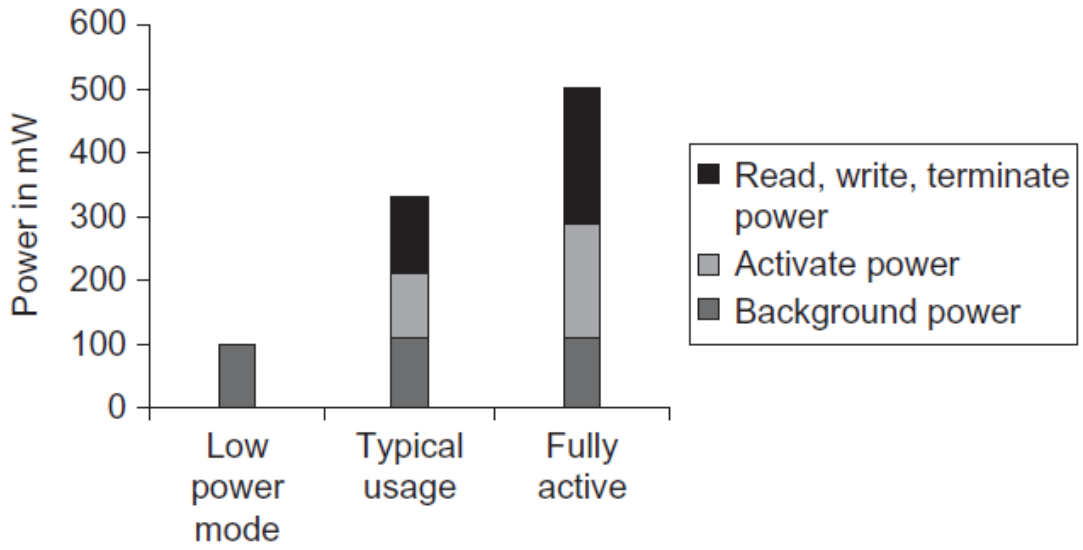


Fig. 5. Power consumption for a DDR3 SDRAM operating under three conditions: low-power (shutdown) mode, typical system mode (DRAM is active 30% of the time for reads and 15% for writes), and fully active mode, where the DRAM is continuously reading or writing

Graphics Data RAMs

- GDRAM (Graphics DRAM) & GSDRAM (Graphics Synchronous DRAM)
  1. are a special class of DRAMs based on SDRAM designs but tailored for handling the higher bandwidth demands of graphics processing units
  2. normally connect directly to the GPU and are attached by soldering them to the board, unlike DRAMs, which are normally arranged in an expandable array of DIMMs
- GDDR5 is based on DDR3 with earlier GDDRs based on DDR2
  1. has wider interfaces: 32-bits versus 4, 8, or 16 in current designs
  2. has a higher maximum clock rate on the data pins
  3. runs at two to five times the bandwidth per DRAM versus DDR3 DRAMs

Flash Memory

- Flash memory is a type of EEPROM (Electronically Erasable Programmable Readonly Memory
- Reads to Flash are sequential and read an entire page, which can be 512 bytes, 2 KiB, or 4 KiB)
- Flash memory must be erased before it is overwritten
- Flash memory is nonvolatile
- Flash memory limits the number of times that any given block can be written
- High-density NAND Flash is cheaper than SDRAM but more expensive than disks

Ten Advanced Optimizations of Cache Performance

Five categories:

1. Reducing the hit time
2. Increasing cache bandwidth
3. Reducing the miss penalty
4. Reducing the miss rate
5. Reducing the miss penalty or miss rate via parallelism

Ten Advanced Optimizations of (

Cache size (KiB)	Degree associative	Total miss rate	Miss rate components (relative percent) (sum = 100% of total miss rate)					
			Compulsory		Capacity		Conflict	
4	1-way	0.098	0.0001	0.1%	0.070	72%	0.027	28%
4	2-way	0.076	0.0001	0.1%	0.070	93%	0.005	7%
4	4-way	0.071	0.0001	0.1%	0.070	99%	0.001	1%
4	8-way	0.071	0.0001	0.1%	0.070	100%	0.000	0%
8	1-way	0.068	0.0001	0.1%	0.044	65%	0.024	35%
8	2-way	0.049	0.0001	0.1%	0.044	90%	0.005	10%
8	4-way	0.044	0.0001	0.1%	0.044	99%	0.000	1%
8	8-way	0.044	0.0001	0.1%	0.044	100%	0.000	0%
16	1-way	0.049	0.0001	0.1%	0.040	82%	0.009	17%
16	2-way	0.041	0.0001	0.2%	0.040	98%	0.001	2%
16	4-way	0.041	0.0001	0.2%	0.040	99%	0.000	0%
16	8-way	0.041	0.0001	0.2%	0.040	100%	0.000	0%
32	1-way	0.042	0.0001	0.2%	0.037	89%	0.005	11%
32	2-way	0.038	0.0001	0.2%	0.037	99%	0.000	0%
32	4-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
32	8-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
64	1-way	0.037	0.0001	0.2%	0.028	77%	0.008	23%
64	2-way	0.031	0.0001	0.2%	0.028	91%	0.003	9%
64	4-way	0.030	0.0001	0.2%	0.028	95%	0.001	4%
64	8-way	0.029	0.0001	0.2%	0.028	97%	0.001	2%
128	1-way	0.021	0.0001	0.3%	0.019	91%	0.002	8%
128	2-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	4-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	8-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
256	1-way	0.013	0.0001	0.5%	0.012	94%	0.001	6%
256	2-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	4-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	8-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
512	1-way	0.008	0.0001	0.8%	0.005	66%	0.003	33%
512	2-way	0.007	0.0001	0.9%	0.005	71%	0.002	28%
512	4-way	0.006	0.0001	1.1%	0.005	91%	0.000	8%
512	8-way	0.006	0.0001	1.1%	0.005	95%	0.000	4%

Fig.6. Total miss rate for each size cache and percentage of each according to the three C's.

Ten Advanced Optimizations of Cache Performance

1. Small and Simple First-Level Caches to Reduce Hit Time and Power
2. Way Prediction to Reduce Hit Time
3. Pipelined Access and Multibanked Caches to Increase Bandwidth
4. Nonblocking Caches to Increase Cache Bandwidth
5. Critical Word First and Early Restart to Reduce Miss Penalty
6. Merging Write Buffer to Reduce Miss Penalty
7. Compiler Optimizations to Reduce Miss Rate
8. Hardware Prefetching of Instructions and Data to Reduce Miss Penalty or Miss Rate
9. Compiler-Controlled Prefetching to Reduce Miss Penalty or Miss Rate
10. Using HBM to Extend the Memory Hierarchy

Small & Simple First-Level Caches to Reduce Hit Time and Power

- The pressure of both a fast clock cycle and power limitations encourages limited size for first-level caches

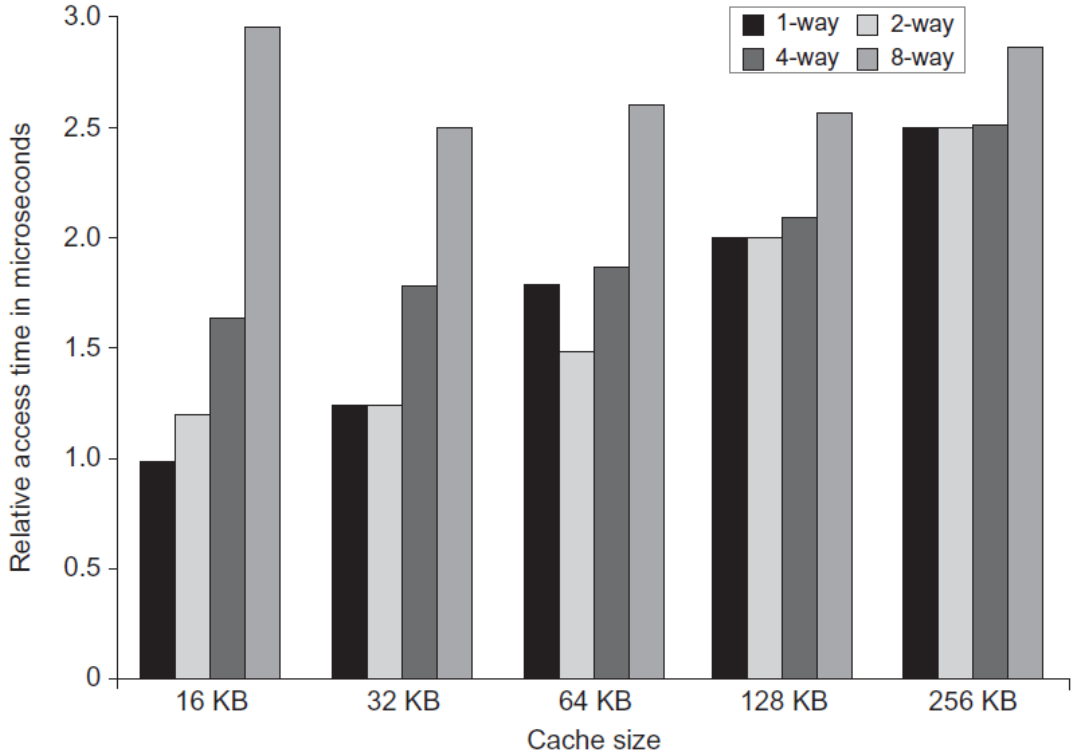


Fig. 6. Relative access times generally increase as cache size and associativity are increased

Cache size (KiB)	Degree associative	Total miss rate	Miss rate components (relative percent) (sum = 100% of total miss rate)					
			Compulsory		Capacity		Conflict	
4	1-way	0.098	0.0001	0.1%	0.070	72%	0.027	28%
4	2-way	0.076	0.0001	0.1%	0.070	93%	0.005	7%
4	4-way	0.071	0.0001	0.1%	0.070	99%	0.001	1%
4	8-way	0.071	0.0001	0.1%	0.070	100%	0.000	0%
8	1-way	0.068	0.0001	0.1%	0.044	65%	0.024	35%
8	2-way	0.049	0.0001	0.1%	0.044	90%	0.005	10%
8	4-way	0.044	0.0001	0.1%	0.044	99%	0.000	1%
8	8-way	0.044	0.0001	0.1%	0.044	100%	0.000	0%
16	1-way	0.049	0.0001	0.1%	0.040	82%	0.009	17%
16	2-way	0.041	0.0001	0.2%	0.040	98%	0.001	2%
16	4-way	0.041	0.0001	0.2%	0.040	99%	0.000	0%
16	8-way	0.041	0.0001	0.2%	0.040	100%	0.000	0%
32	1-way	0.042	0.0001	0.2%	0.037	89%	0.005	11%
32	2-way	0.038	0.0001	0.2%	0.037	99%	0.000	0%
32	4-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
32	8-way	0.037	0.0001	0.2%	0.037	100%	0.000	0%
64	1-way	0.037	0.0001	0.2%	0.028	77%	0.008	23%
64	2-way	0.031	0.0001	0.2%	0.028	91%	0.003	9%
64	4-way	0.030	0.0001	0.2%	0.028	95%	0.001	4%
64	8-way	0.029	0.0001	0.2%	0.028	97%	0.001	2%
128	1-way	0.021	0.0001	0.3%	0.019	91%	0.002	8%
128	2-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	4-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
128	8-way	0.019	0.0001	0.3%	0.019	100%	0.000	0%
256	1-way	0.013	0.0001	0.5%	0.012	94%	0.001	6%
256	2-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	4-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
256	8-way	0.012	0.0001	0.5%	0.012	99%	0.000	0%
512	1-way	0.008	0.0001	0.8%	0.005	66%	0.003	33%
512	2-way	0.007	0.0001	0.9%	0.005	71%	0.002	28%
512	4-way	0.006	0.0001	1.1%	0.005	91%	0.000	8%
512	8-way	0.006	0.0001	1.1%	0.005	95%	0.000	4%

Fig. 7. Total miss rate for each size cache and percentage of each according to the three C's.

Example

- Using Fig.6 and Fig.7 to determine whether a 32 KiB four-way set associative L1 cache has a faster memory access time than a 32 KiB two-way set associative L1 cache. Assume the miss penalty to L2 is 15 times the access time for the faster L1 cache. Ignore misses beyond L2. Which has the faster average memory access time?

- Ans: Let the access time for the 2-way set associative cache be 1. Then, for the 2-way cache

$$\begin{aligned}\text{Average memory access time}_{\text{2-way}} &= \text{Hit time} + \text{Miss rate} \times \text{Miss penalty} \\ &= 1 + 0.038 \times 15 = 1.38\end{aligned}$$

- For the four-way cache, the access time is 1.4 times longer. The elapsed time of the miss penalty is 15/1.4 = 10.1. Assume 10 for simplicity:

$$\begin{aligned}\text{Average memory access time}_{\text{4-way}} &= \text{Hit time}_{\text{2-way}} \times 1.4 + \text{Miss rate} \times \text{Miss penalty} \\ &= 1.4 + 0.037 \times 10 = 1.77\end{aligned}$$

Compiler Optimizations to Reduce Miss Rate

Loop Interchange:

- Assuming the arrays do not fit in the cache, this technique reduces misses by improving spatial locality; reordering maximizes use of data in a cache block before they are discarded

```
/* Before */
for (j = 0; j < 100; j = j + 1)
        for (i = 0; i < 5000; i = i + 1)
                x[i][j] = 2 * x[i][j];
/* After */
for (i = 0; i < 5000; i = i + 1)
        for (j = 0; j < 100; j = j + 1)
                x[i][j] = 2 * x[i][j];
```

Compiler Optimizations to Reduce Miss Rate

Blocking:

- This optimization improves temporal locality to reduce misses

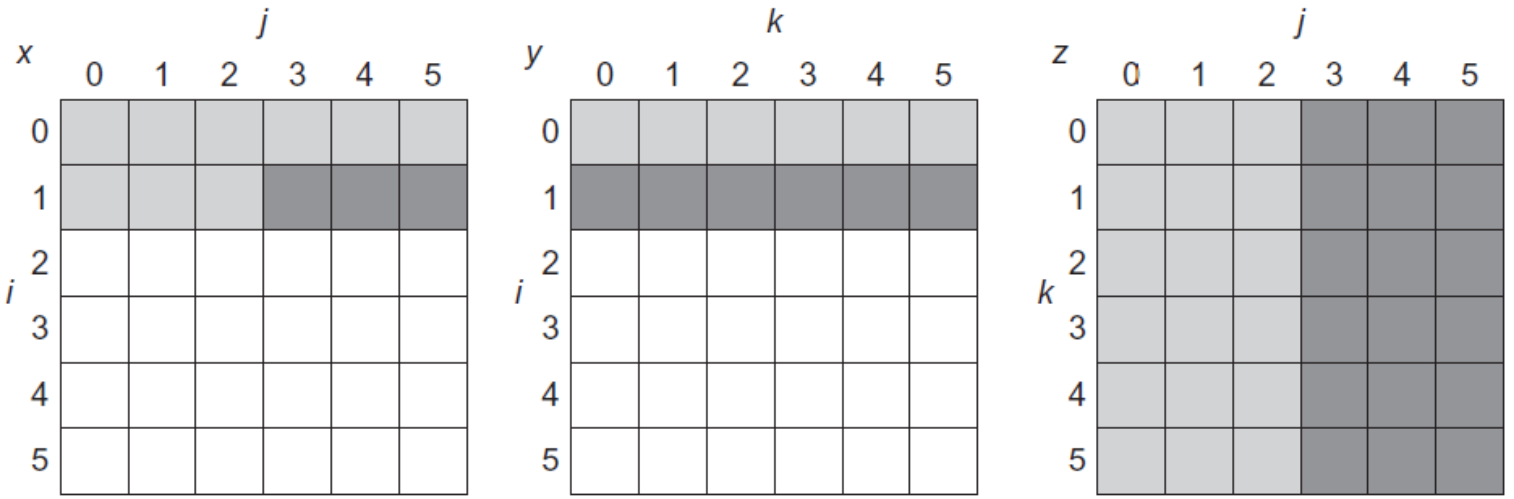


Fig. 8. A snapshot of the three arrays x, y, and z when N = 6 and I = 1. The age of accesses to the array elements is indicated by shade: white means not yet touched, light means older accesses, and dark means newer accesses. The elements of y and z are read repeatedly to calculate new elements of x. The variables i, j, and k are shown along the rows or columns used to access the arrays

Example

Blocking:

- Instead of operating on entire rows or columns of an array, blocked algorithms operate on submatrices or blocks. The goal is to maximize accesses to the data loaded into the cache before the data are replaced

```
/* Before */
for (i = 0; i < N; i = i + 1)
      for (j = 0; j < N; j = j + 1)
           {r = 0;
            for (k = 0; k < N; k = k + 1)
                 r = r + y[i][k]*z[k][j];
            x[i][j] = r;
           };
```

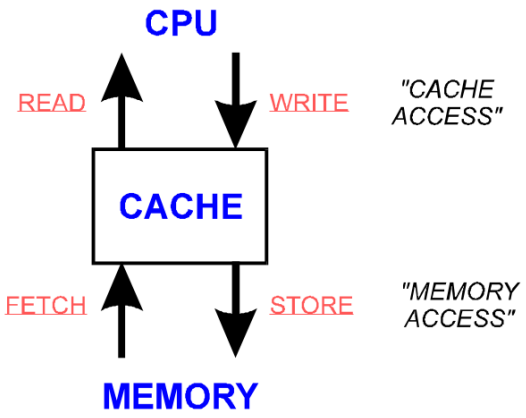
```
/* After */
for (jj = 0; jj < N; jj = jj + B)
for (kk = 0; kk < N; kk = kk + B)
for (i = 0; i < N; i = i + 1)
      for (j = jj; j < min(jj + B,N); j = j + 1)
          {r = 0;
           for (k = kk; k < min(kk + B,N); k = k + 1)
                r = r + y[i][k]*z[k][j];
           x[i][j] = x[i][j] + r;
          };
```

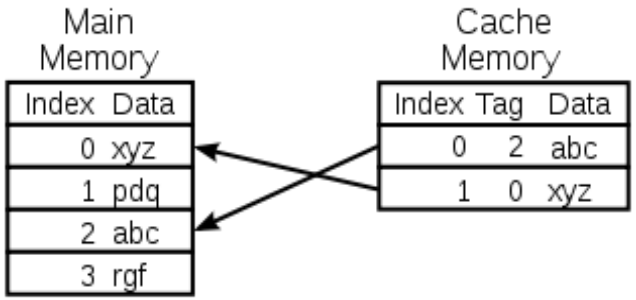
Cache Optimization Summary

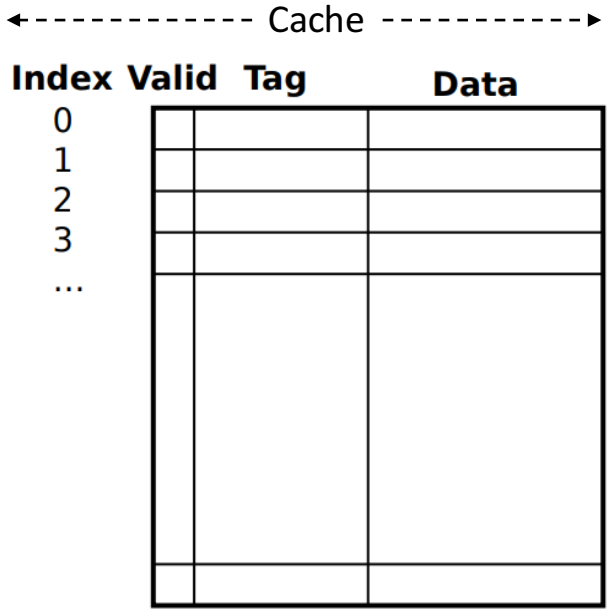
Technique	Hit time	Band-width	Miss penalty	Miss rate	Power consumption	Hardware cost/ complexity	Comment
Small and simple caches	+			–	+	0	Trivial; widely used
Way-predicting caches	+				+	1	Used in Pentium 4
Pipelined & banked caches	–	+				1	Widely used
Nonblocking caches		+	+			3	Widely used
Critical word first and early restart			+			2	Widely used
Merging write buffer			+			1	Widely used with write through
Compiler techniques to reduce cache misses				+		0	Software is a challenge, but many compilers handle common linear algebra calculations
Hardware prefetching of instructions and data			+	+	–	2 instr., 3 data	Most provide prefetch instructions; modern high-end processors also automatically prefetch in hardware
Compiler-controlled prefetching			+	+		3	Needs nonblocking cache; possible instruction overhead; in many CPUs
HBM as additional level of cache		+/–	–	+	+	3	Depends on new packaging technology. Effects depend heavily on hit rate improvements

Fig.8. Summary of 10 advanced cache optimizations showing impact on cache performance, power consumption, and complexity

Cache organization





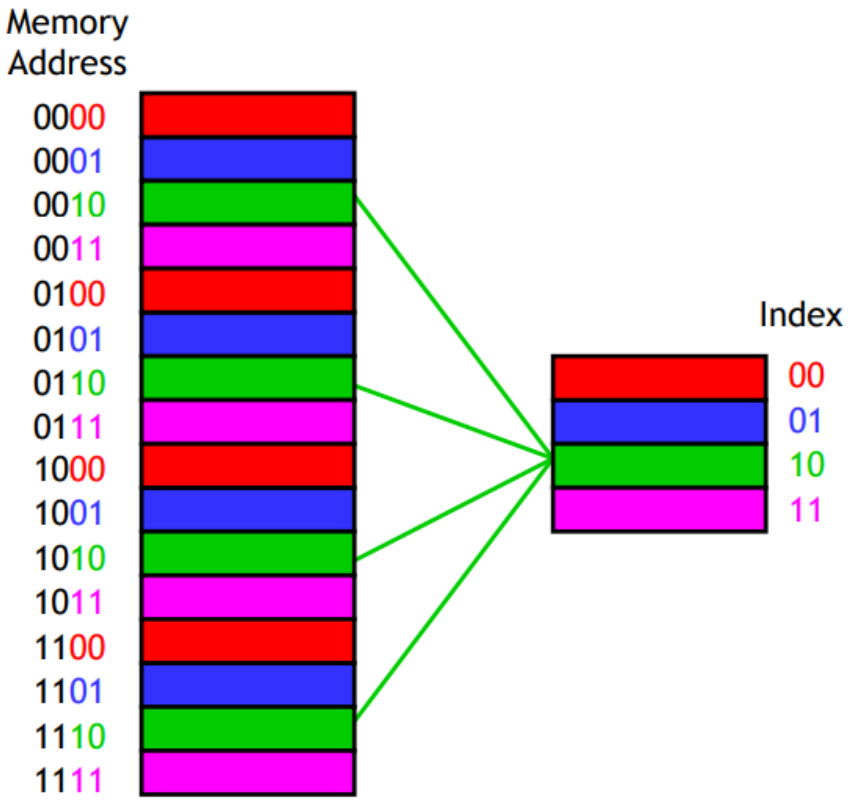


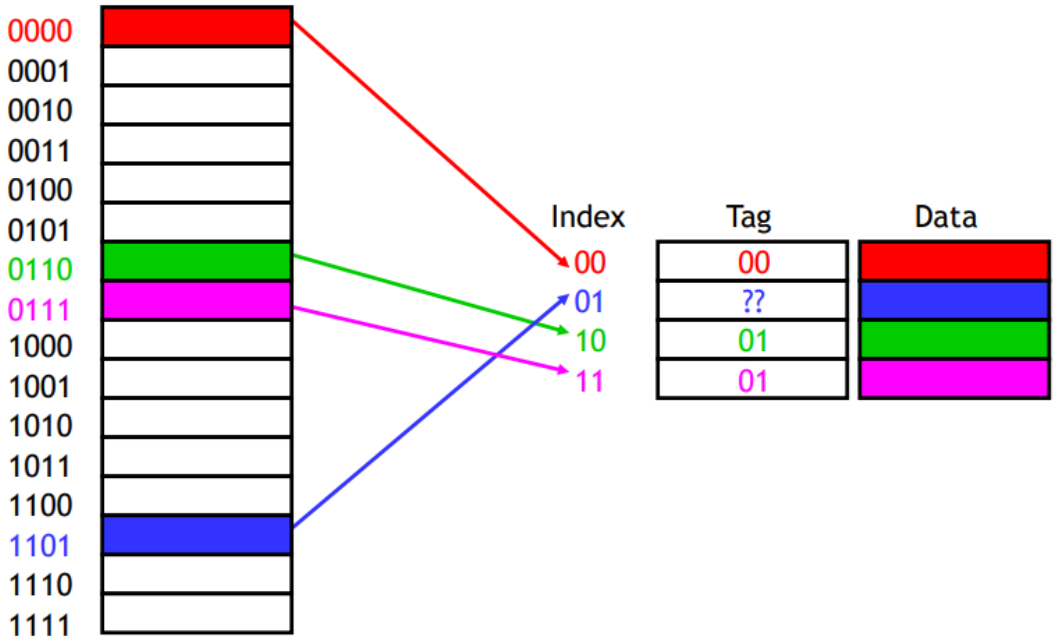
Data: is stored in blocks of main memory
Tag: points to blocks in main memory
Valid: indicates that the data is still valid
Index/set: indicates the position of cache line/block
Offset: indicates the size of a cache line

There are 3 types of cache

1. Direct mapping
2. Set associative/ N-Ways mapping
3. Associative mapping/ Full associative mapping

Direct Mapping





Direct mapping - Example

Memory address

Tag	Index	Offset
010	10	11

	V	Tag	Data 11	10	01	00
00	0					
01	0					
10	1 ~~0~~	010	M(2B)	M(2A)	M(29)	M(28)
11	0					

Cache

Index	V	Tag	Byte1	Byte0
00	~~0~~1	0	5	1
01	~~0~~1	0	7	6
10	~~0~~1	0	7	8
11	~~0~~1	1	2	5

Cache

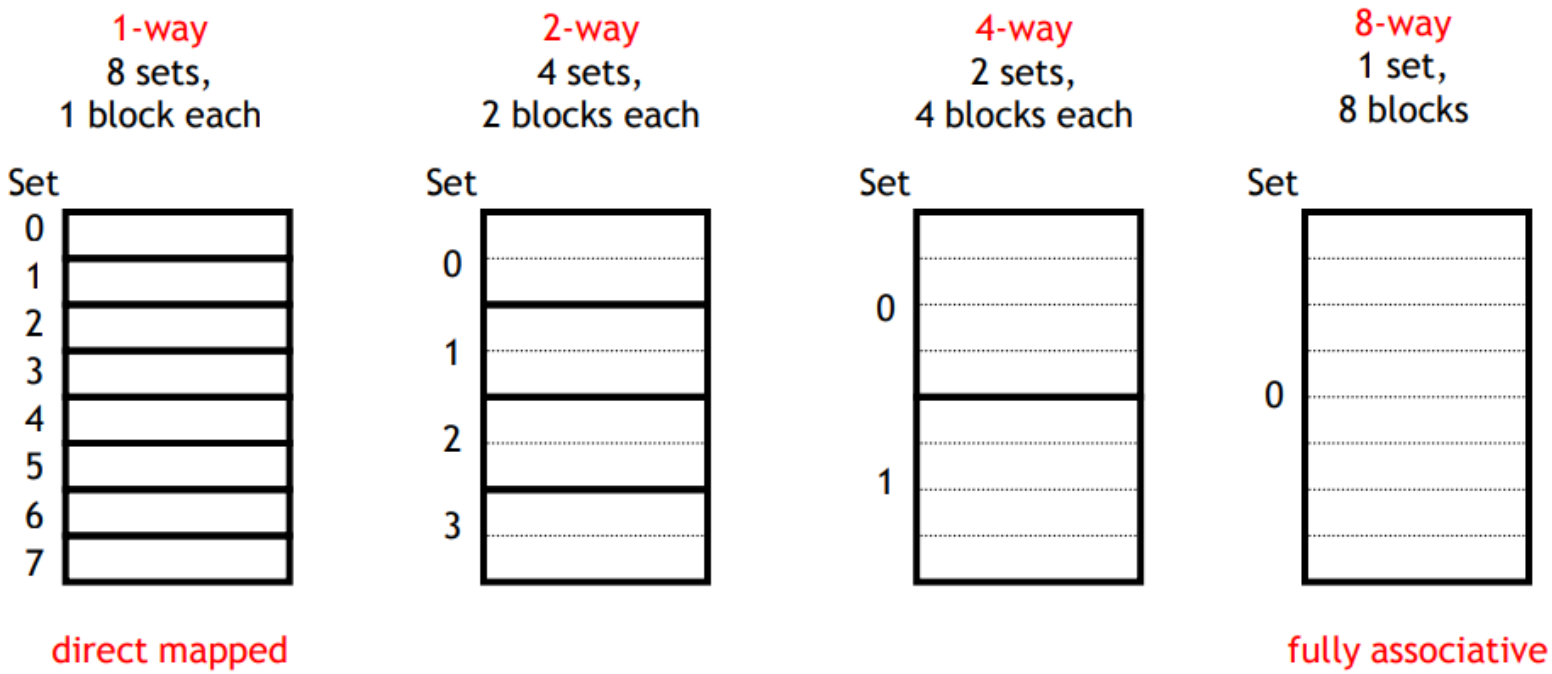
Block	Address	Contents
000	0000	1
	0001	5
001	0010	6
	0011	7
010	0100	8
	0101	7
011	0110	5
	0 111	2
100	1000	2
	1001	9
101	1010	6
	1011	7
110	1100	8
	1101	7
111	1110	5
	1111	2

1	2	1
T ag	Index	Byte offset

Memory address

Cache Memory Mapping Methods

1. Direct mapping
2. Set associative/ N-Ways mapping
3. Associative mapping/ Full associative mapping

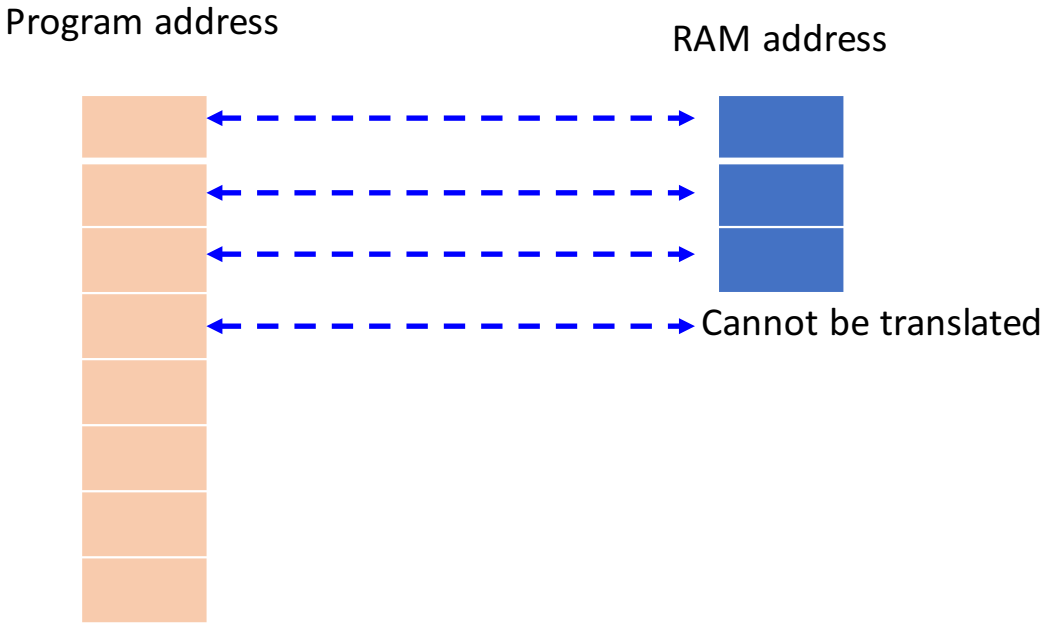


Cache Update Methods

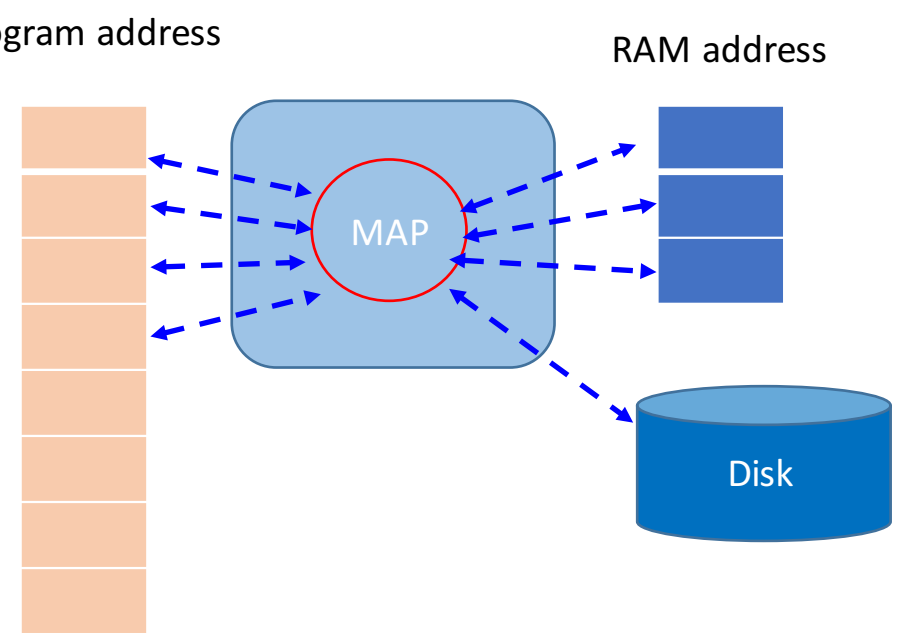
- Write Through:
  - When new information is written to the cache, main memory is also updated
- Buffered Write Through:
  - There is a buffer between cache and main memory, when new information is written to the cache, this information is written into buffer, and the CPU can access this memory before the new information can be written into main memory.
- Write Back (Copy Back):
  - Only the cache is updated, and main memory will be updated when the corresponding cache line is overwritten. In this method each cache line has a dirty bit to indicate if cache line has been modified or not

Virtual Memory

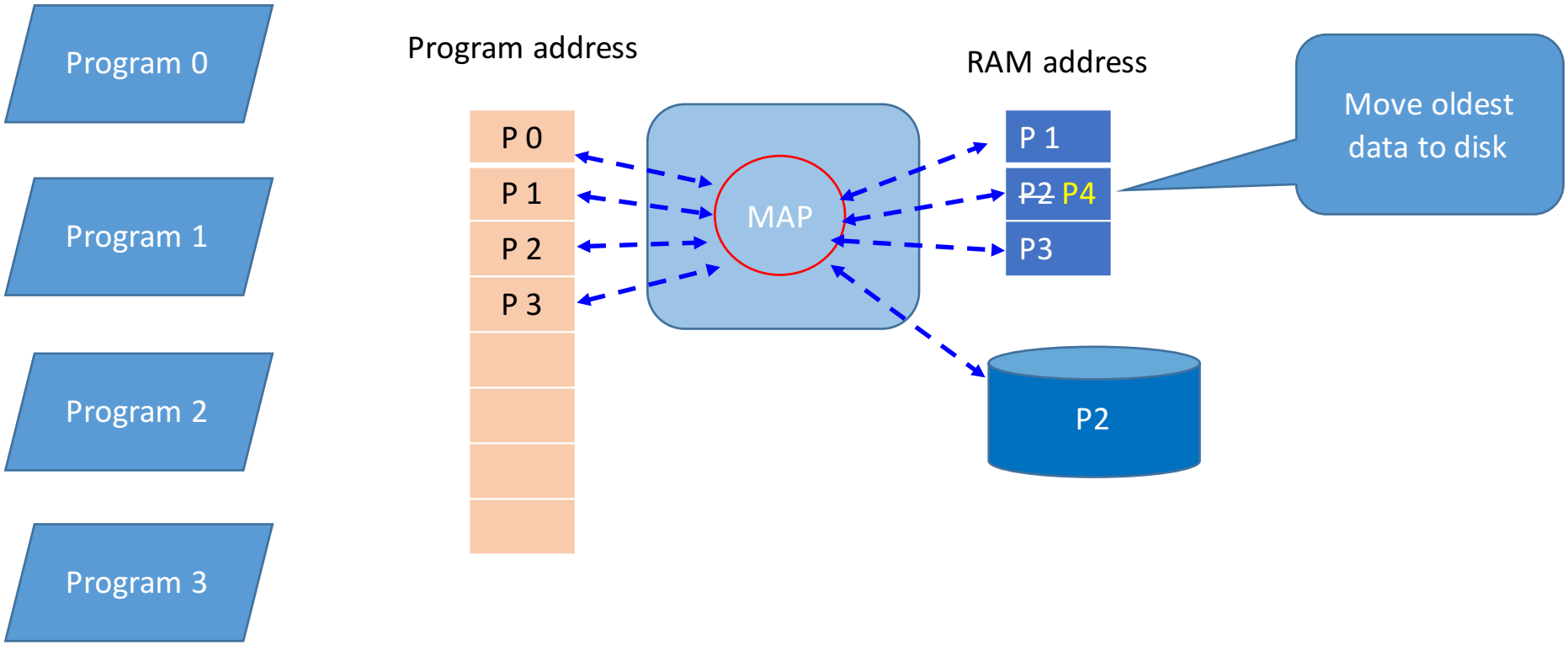
Without Virtual Memory: Program address = RAM address



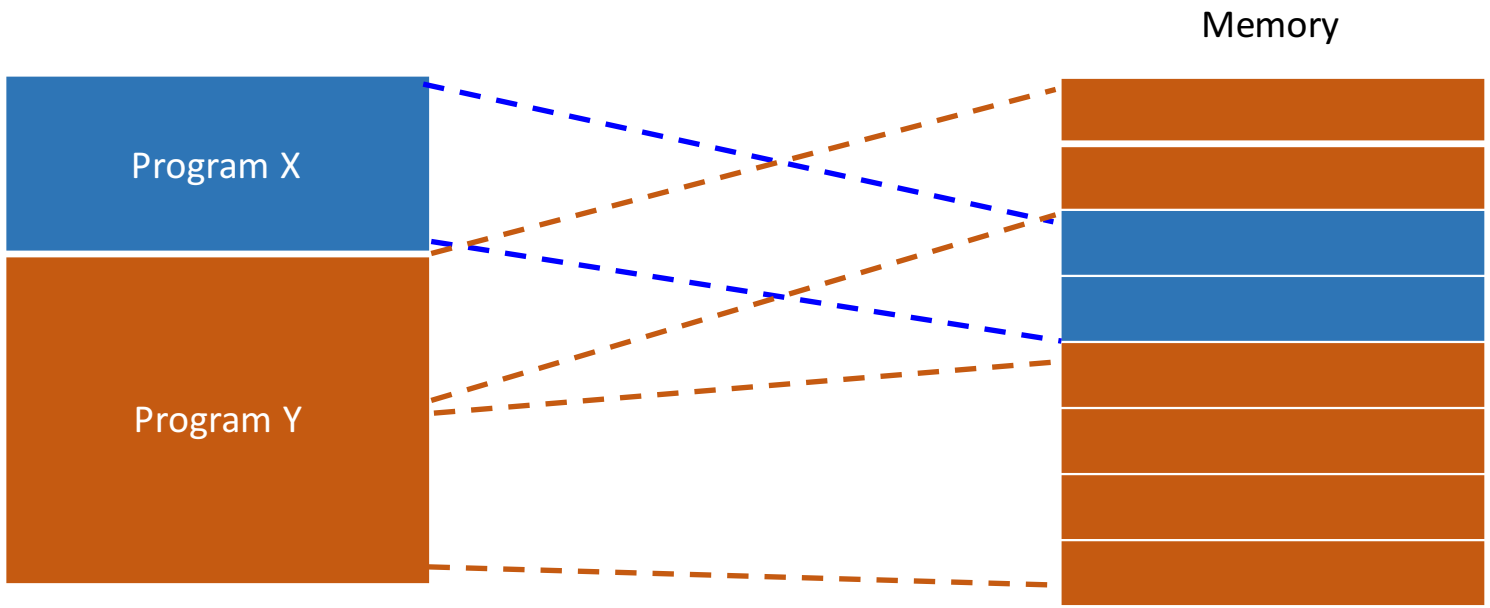
With Virtual Memory: Program address is mapped to RAM



Virtual Memory – Not Enough RAM Memory



Virtual Memory – Memory holes



What you have learned

- An overview of memory hierarchy
- Types of RAM and EEPROM
- Cache optimizations and performance
- Virtual Memory