EPIC
Institute of Technology
Powered by epam

Memory

Course: Systems Architecture

Lecturer: Gleb Lobanov

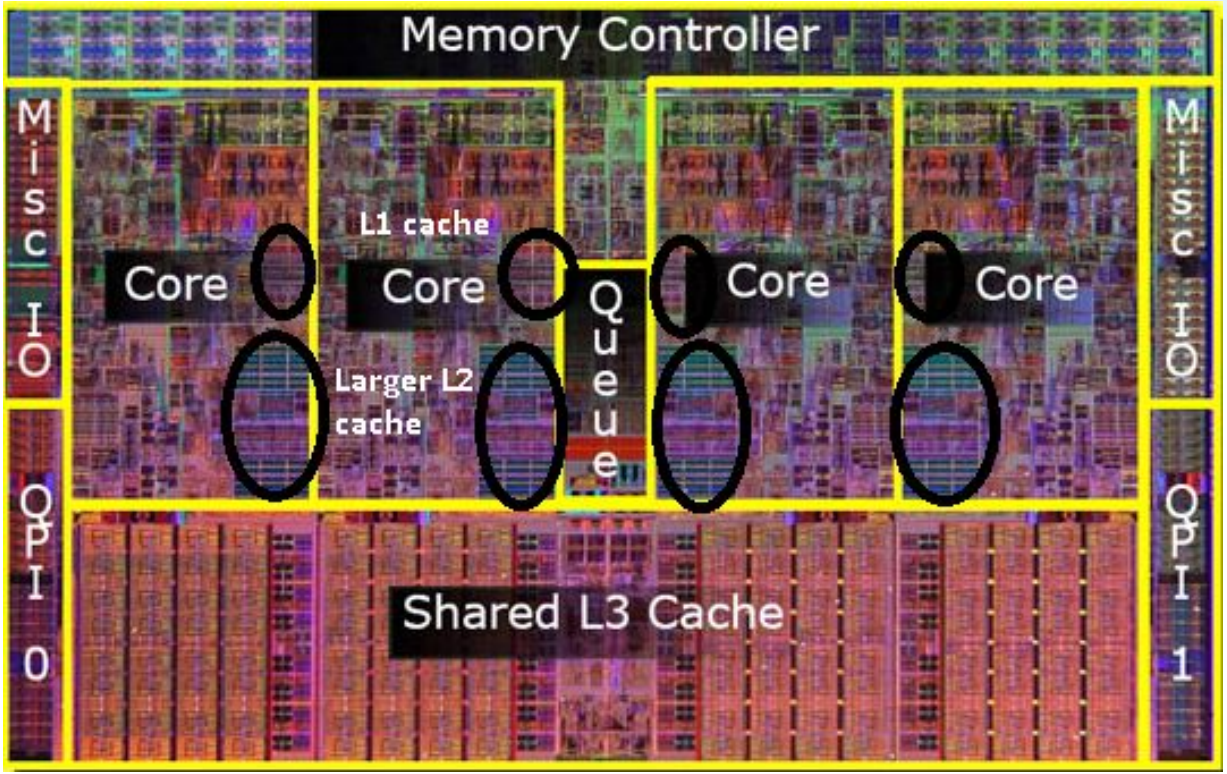
April 1, 2024

EPIC
Institute of Technology
Powered by <epam>

Contents

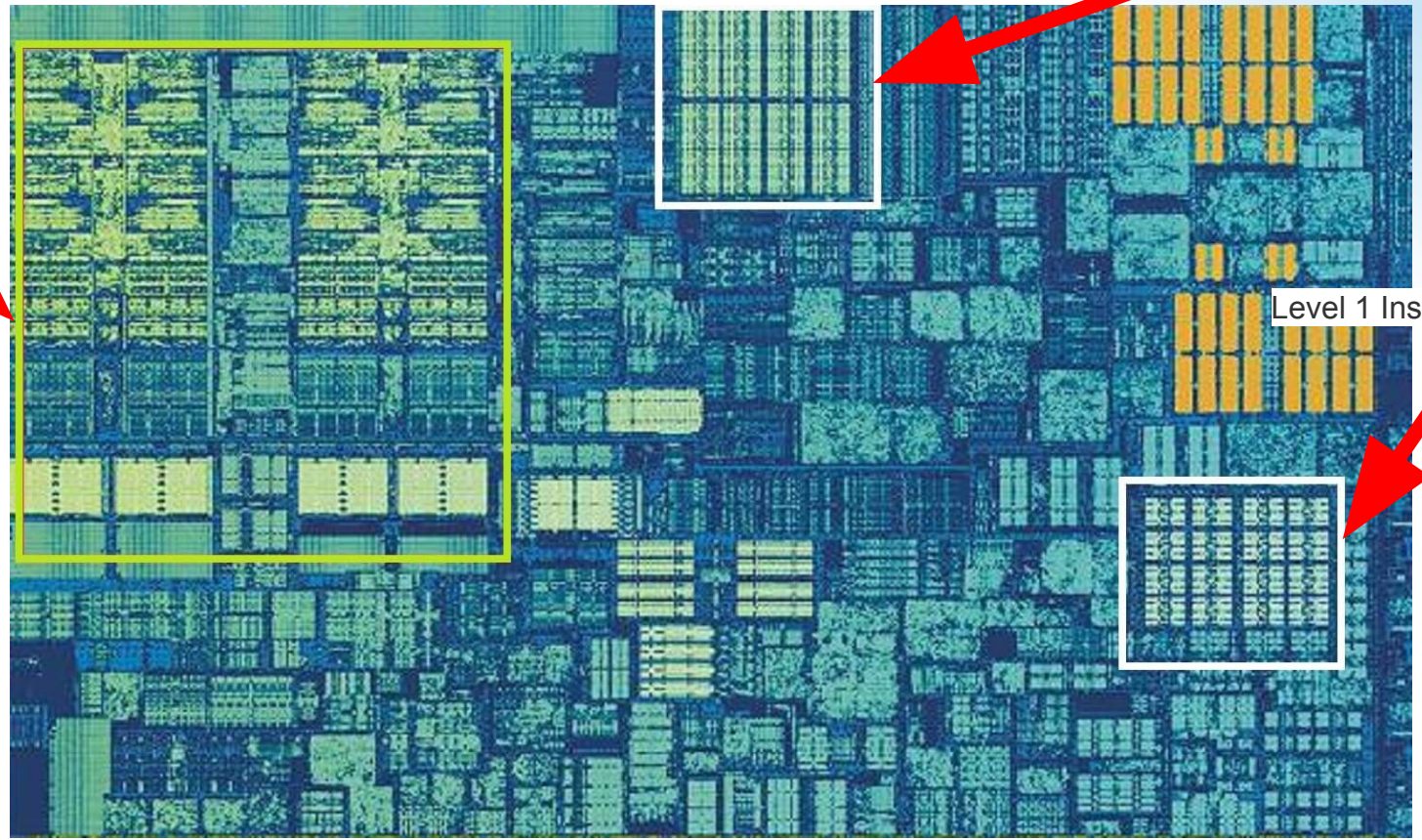
EPIC
Institute of Technology
Powered by epam

Caches



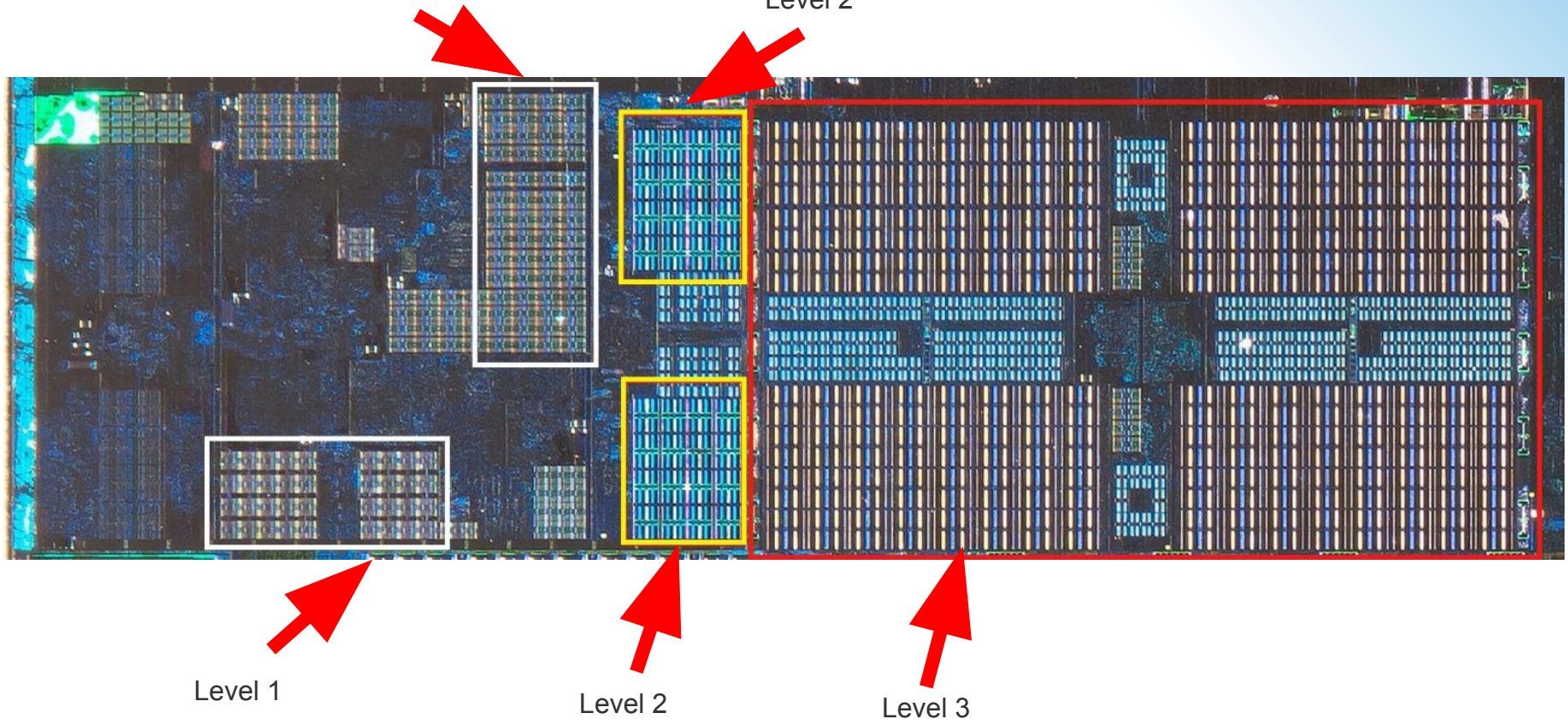
EPIC
Institute of Technology
Powered by <epam>

Caches



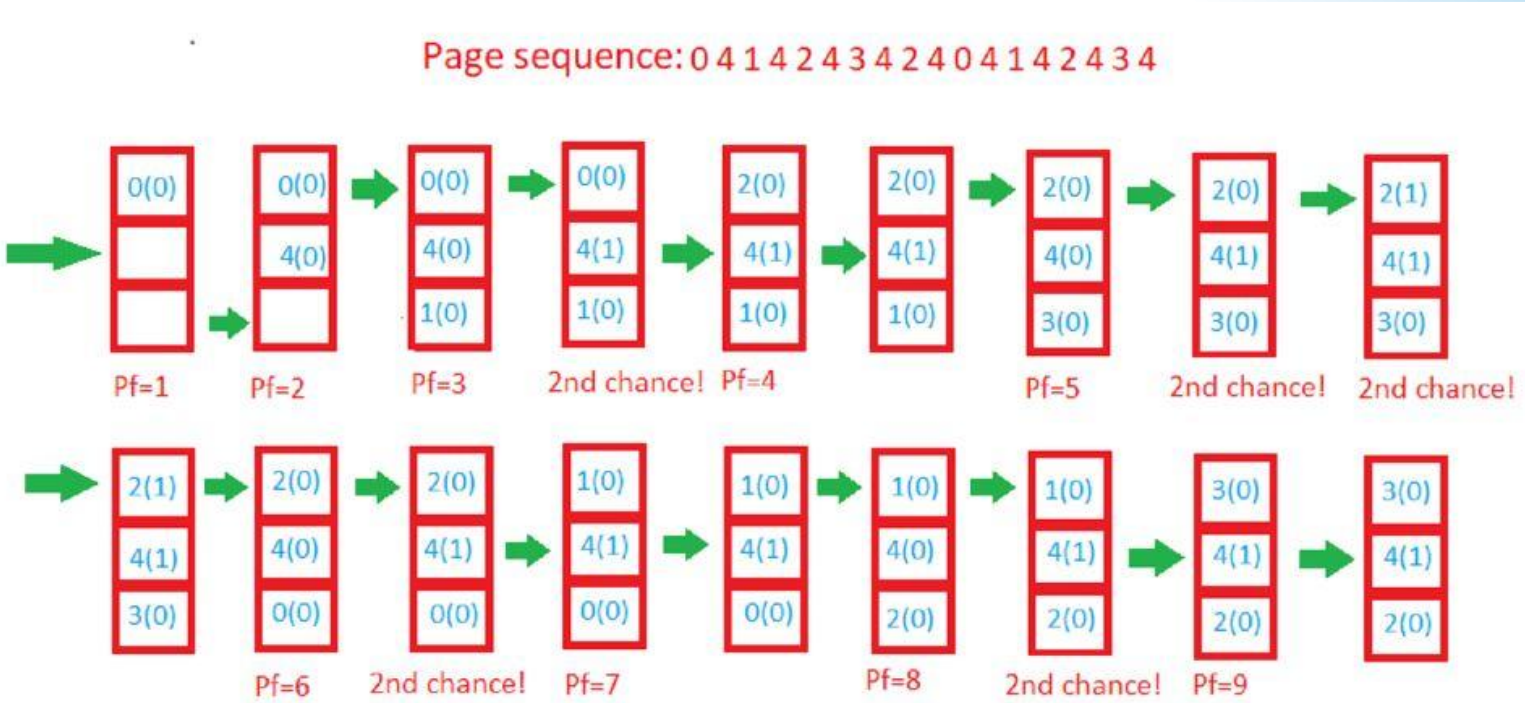
EPIC
Institute of Technology
Powered by epam

Caches



EPIC
Institute of Technology
Powered by epam

Cache in DB



EPIC
Institute of Technology
Powered by epam

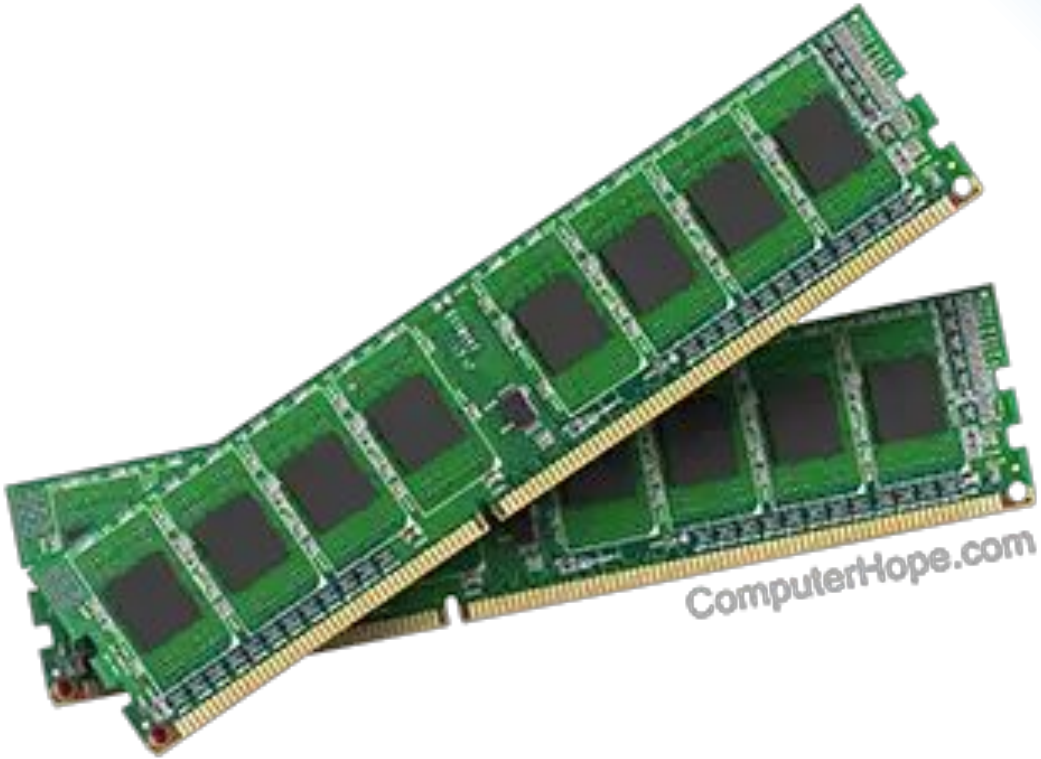
06

RAM

In this section, we will discuss RAM.

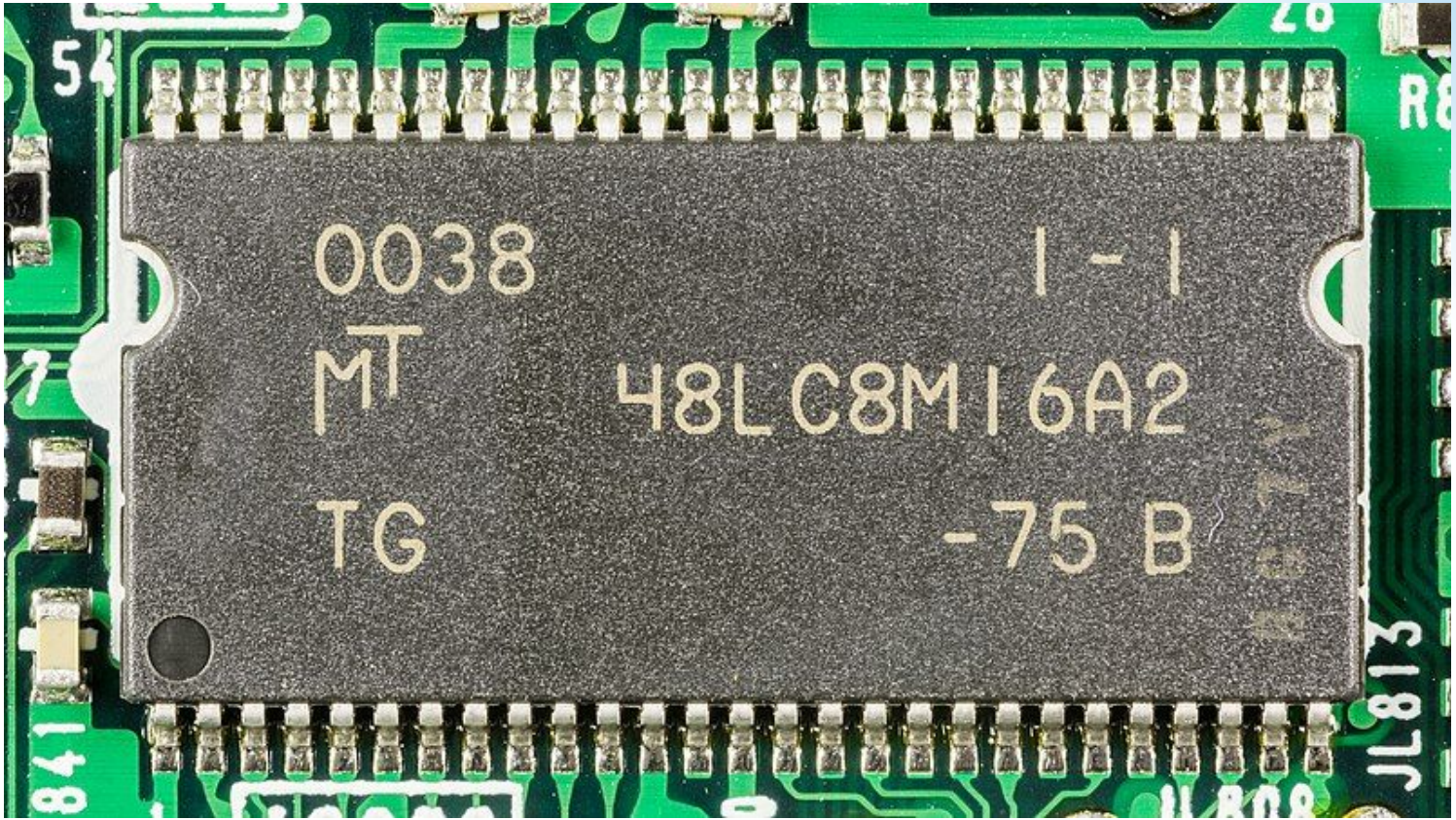
EPIC
Institute of Technology
Powered by epam

RAM



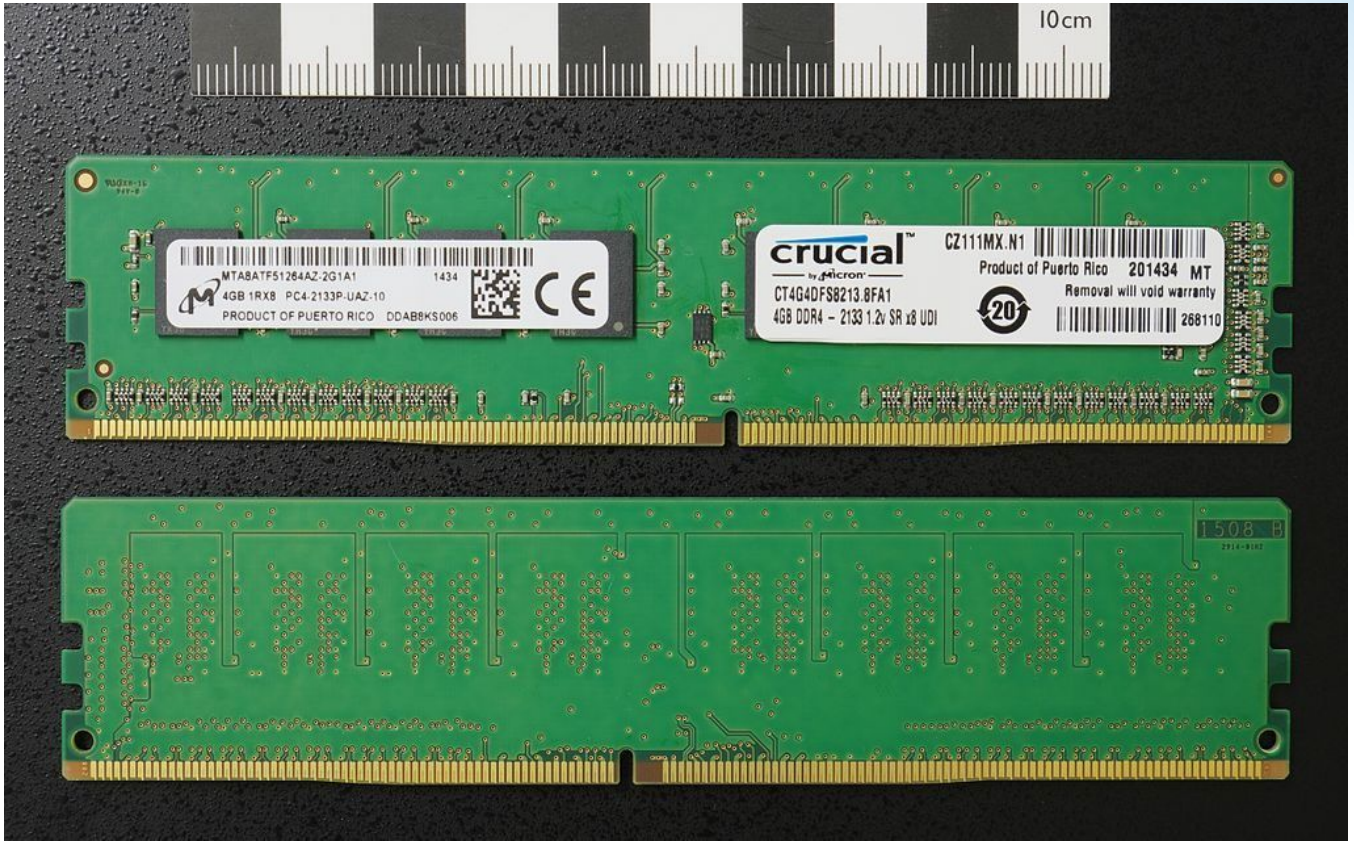
EPIC
Institute of Technology
Powered by epam

SDR



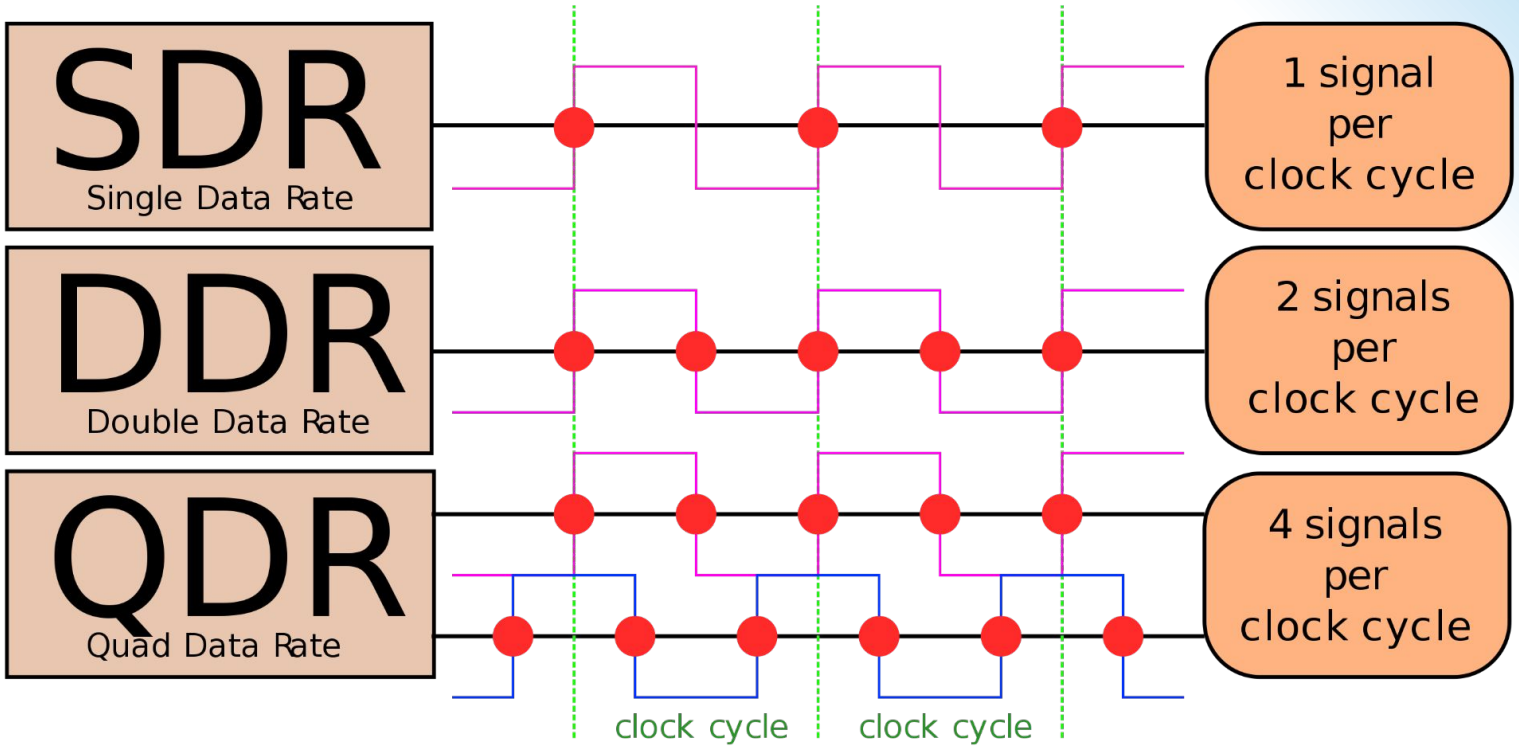
EPIC
Institute of Technology
Powered by epam

DDR



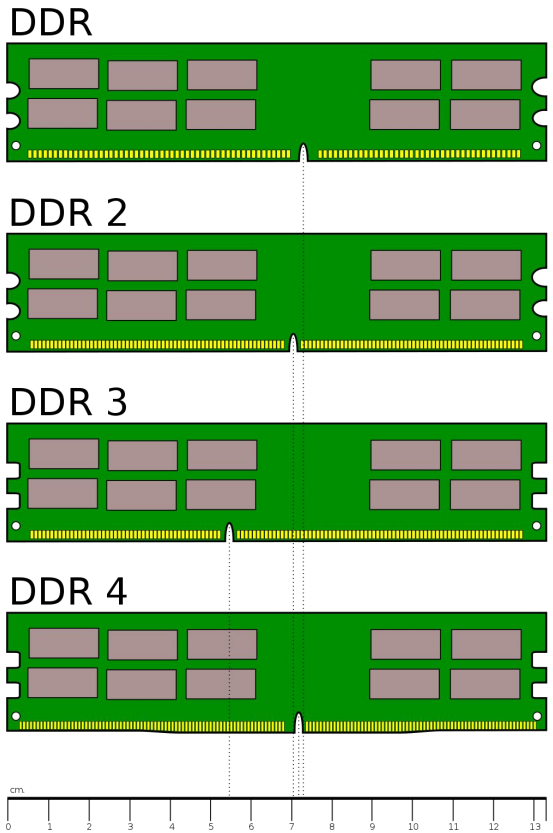
EPIC
Institute of Technology
Powered by epam

How is working



EPIC
Institute of Technology
Powered by <epam>

DDR version



EPIC
Institute of Technology
Powered by <epam>

Example

8th Generation Intel® Core™ i3 Desktop Processors | DDR4-2400 | 64 GB

Sponsored

8GB DDR4-2666 PC4-21300 Non-ECC UDIMM Desktop PC RAM Memory Upgrade

4.5 out of 5 stars ▾ 87

$19.99

Delivery **Mon, Apr 15**

Ships to Serbia

Add to cart

EPIC
Institute of Technology
Powered by epam

Example

11th Generation Intel® Core™ i7 Desktop Processors

DDR4-3200

128 GB

Sponsored

8GB DDR4-2666 PC4-21300 Non-ECC UDIMM Desktop PC RAM Memory Upgrade

4.5 out of 5 stars 87

$19.99

Delivery **Mon, Apr 15**

Ships to Serbia

Add to cart

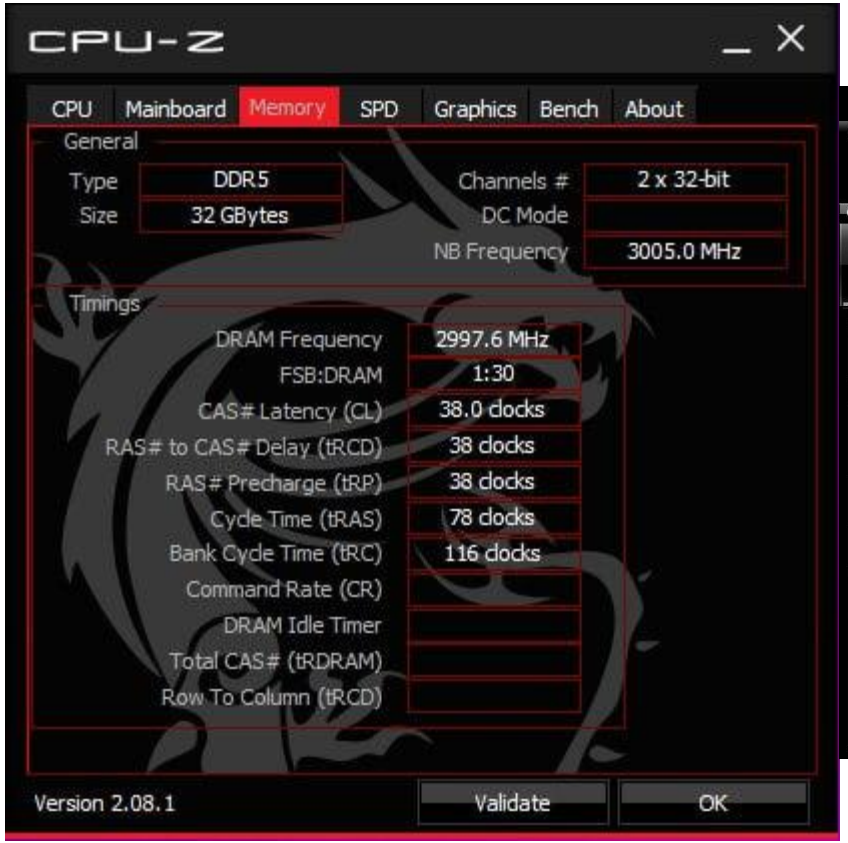
EPIC
Institute of Technology
Powered by <epam>

Example

11th Generation Intel® Core™ i7 Desktop Processors	DDR4-3200	128 GB

EPIC
Institute of Technology
Powered by <epam>

Example



EPIC
Institute of Technology
Powered by <epam>

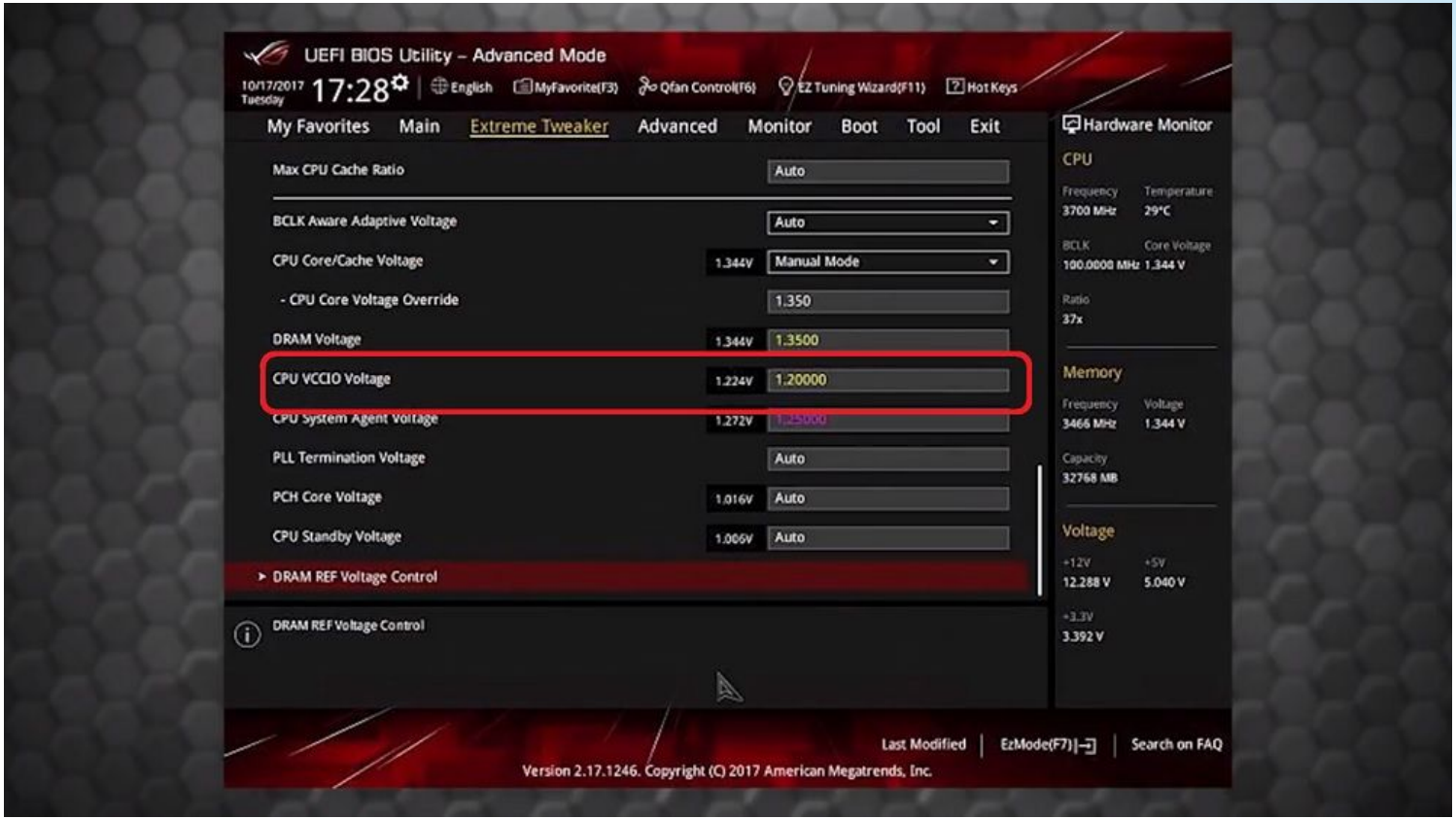
Overclock

EPIC
Institute of Technology
Powered by epam

Result

EPIC
Institute of Technology
Powered by <epam>

Good solution



EPIC
Institute of Technology
Powered by epam

Result

EPIC
Institute of Technology
Powered by epam

Good solution

EPIC
Institute of Technology
Powered by epam

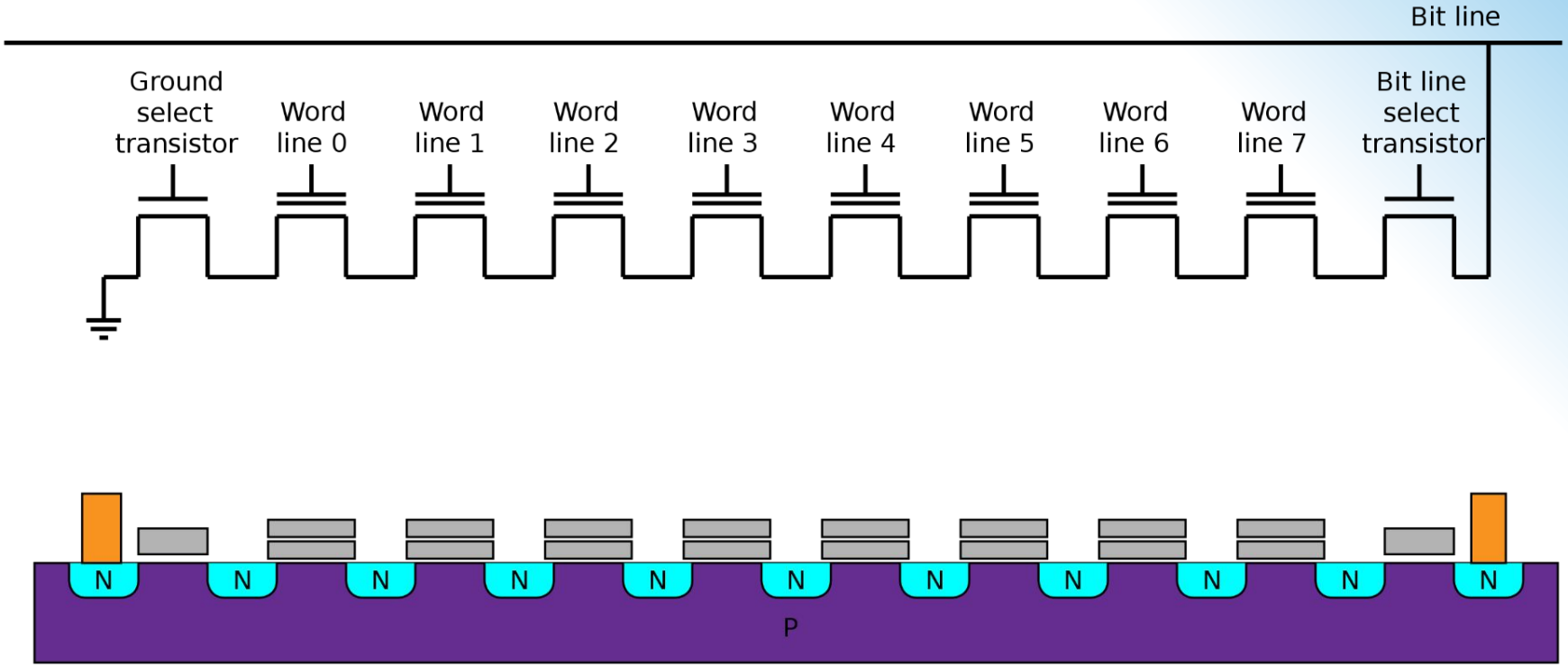
07

SSD

In this section, we will discuss SSD.

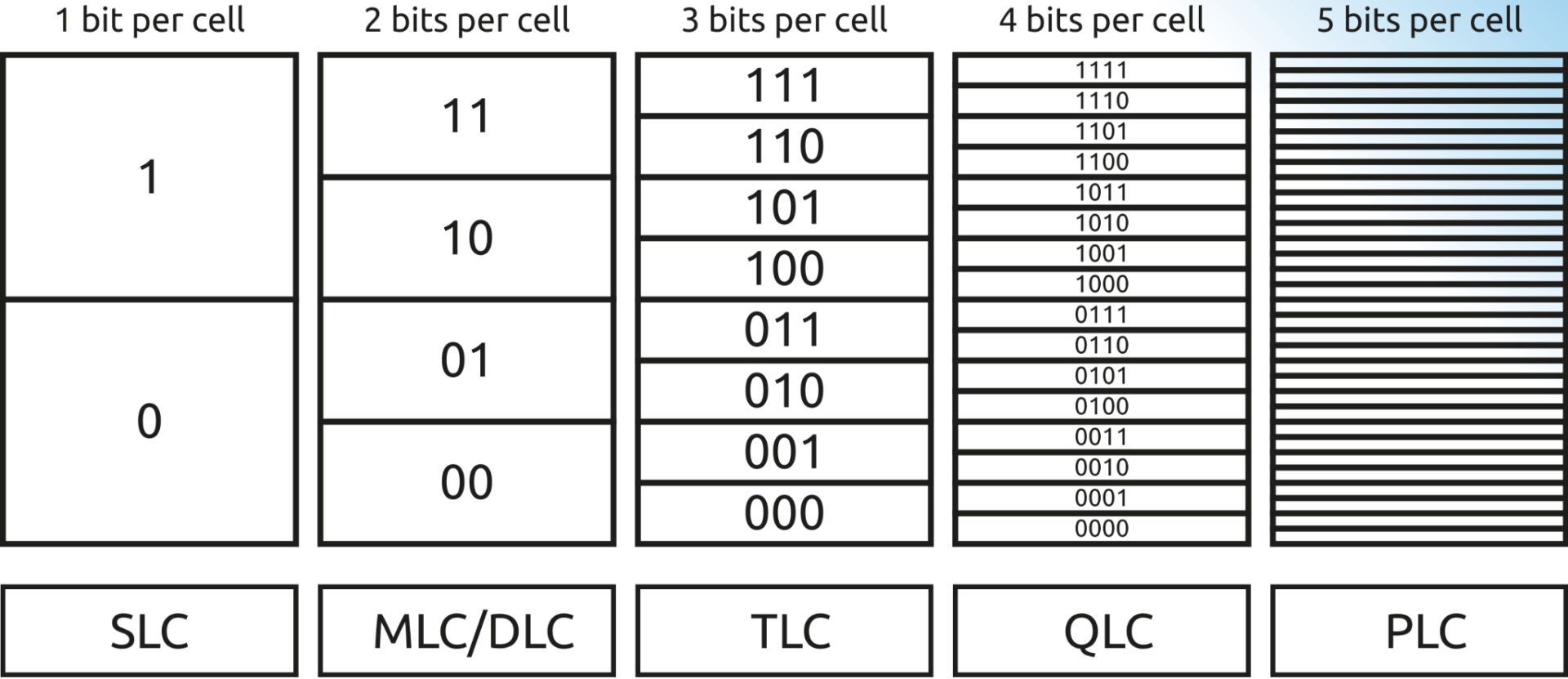
EPIC
Institute of Technology
Powered by epam

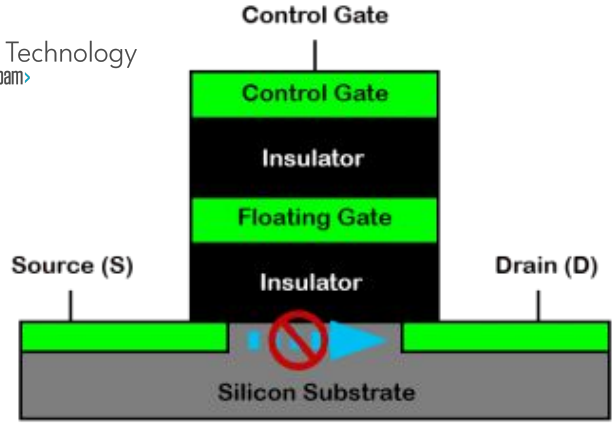
NAND



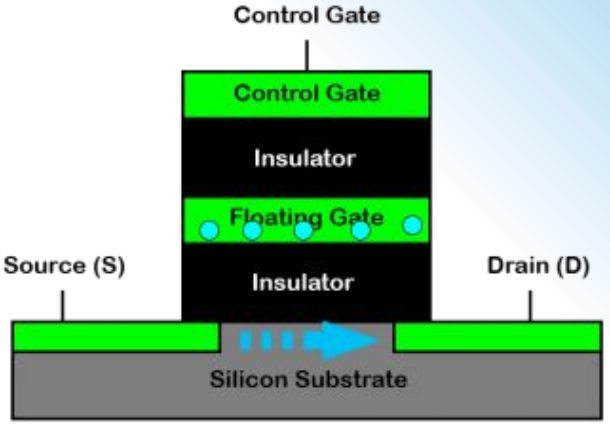
EPIC Institute of Technology
Powered by epam

Cell types of storage

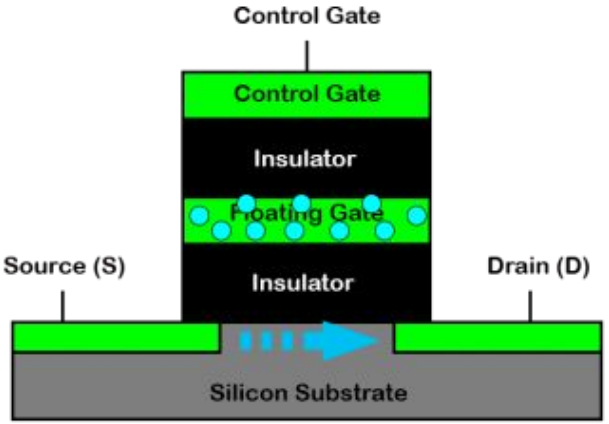




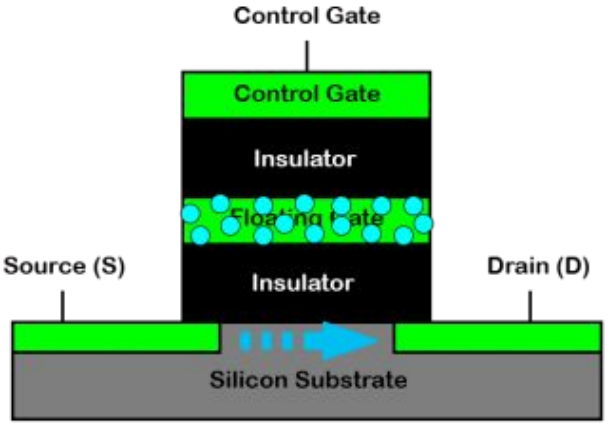
State 1 - No Charge



State 2 - Lightly Charged



State 3 - Medium Charge

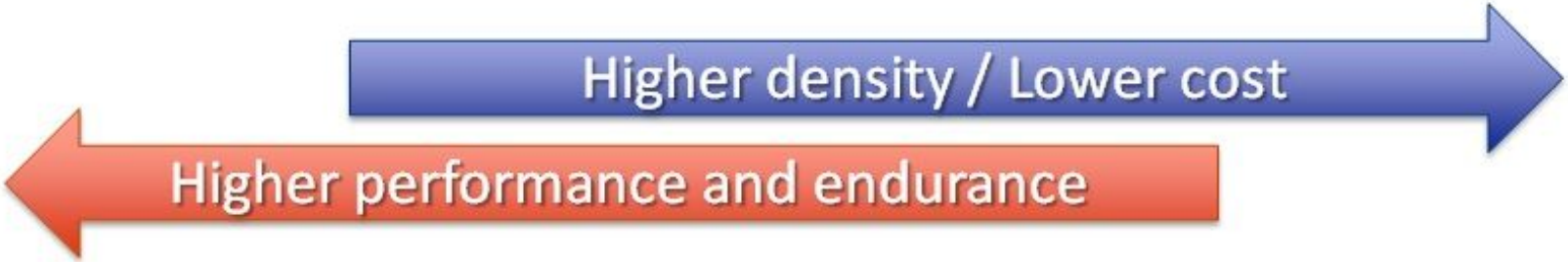


State 4 - Highly Charged

EPIC
Institute of Technology
Powered by epam

Benchmarks

	SLC	MLC	TLC
Bits per cell	1	2	3
P/E Cycles	100,000	3,000	1,000
Read Time	25 µs	50 µs	~75 µs
Program Time	200-300 µs	600-900 µs	~900-1350 µs
Erase Time	1.5-2 ms	3 ms	4.5 ms



Source: http://www.anandtech.com/show/6337/samsung-ssd-840-250gb-review/2

EPIC
Institute of Technology
Powered by epam

07

HDD

In this section, we will discuss HDD.

EPIC
Institute of Technology
Powered by <epam>

HDD

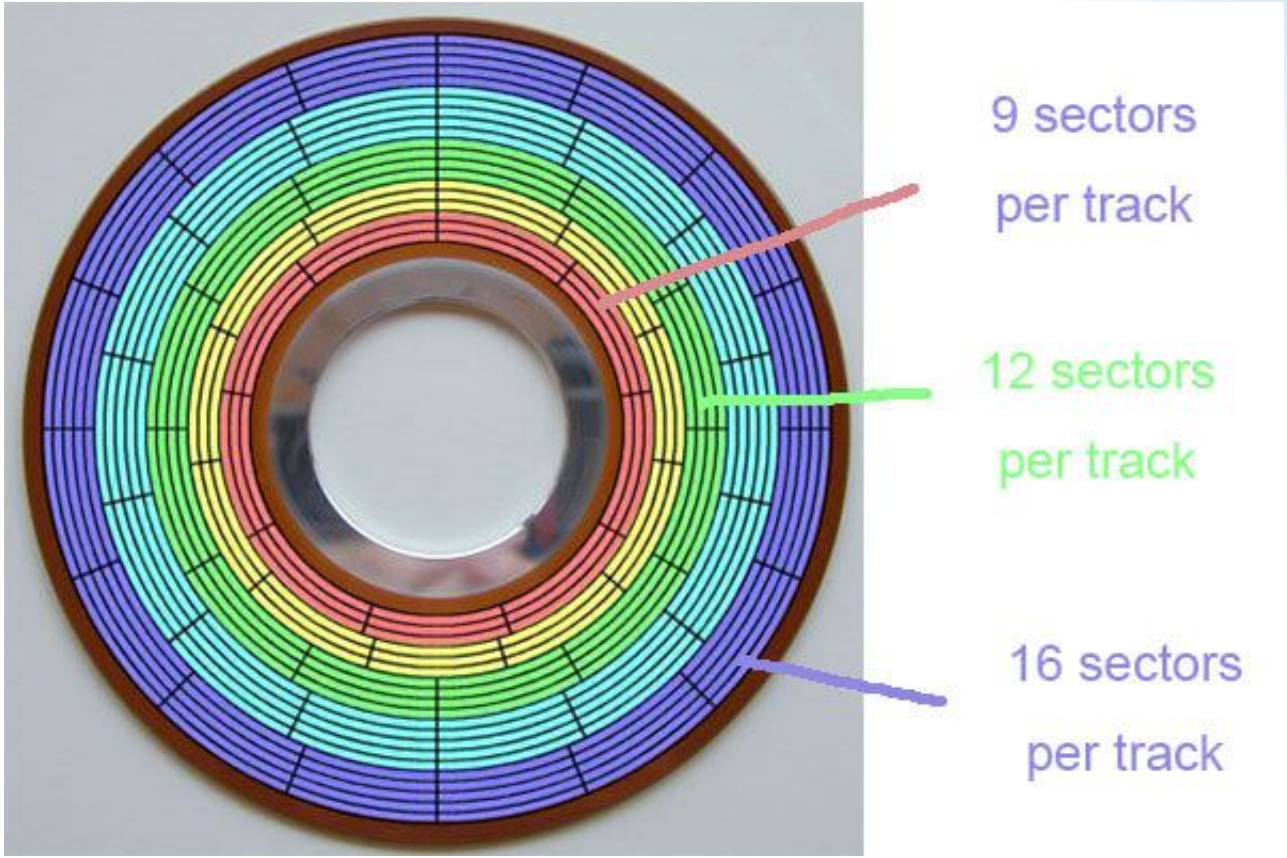
The picture below shows a internal and external view of 2.5-inch HDD:

Internal View

External View

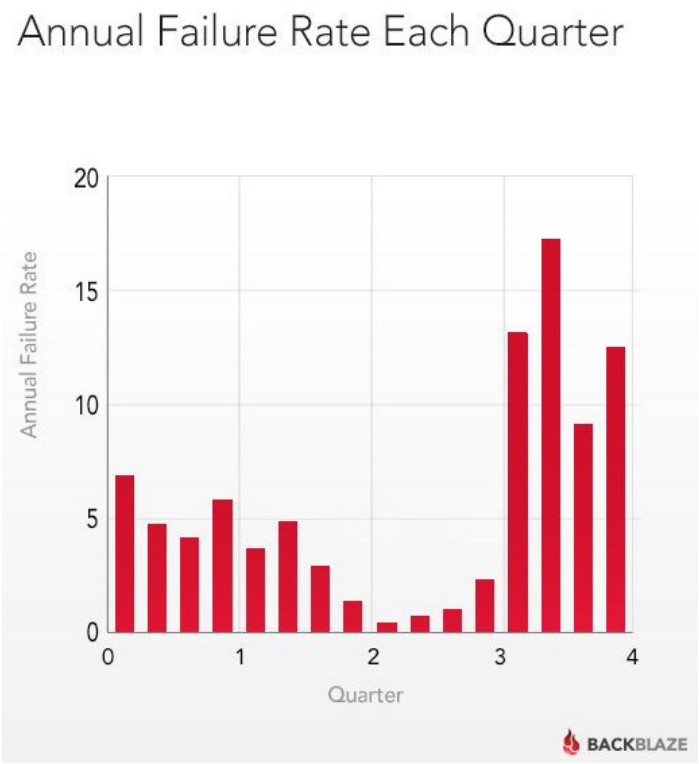
EPIC
Institute of Technology
Powered by epam

HDD division



EPIC
Institute of Technology
Powered by <epam>

Reliability

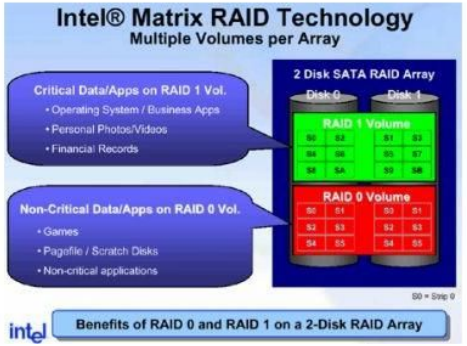


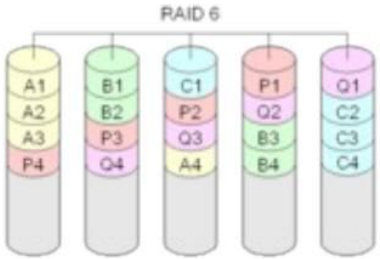
EPIC
Institute of Technology
Powered by <epam>

RAID

EPIC
Institute of Technology
Powered by <epam>

RAID



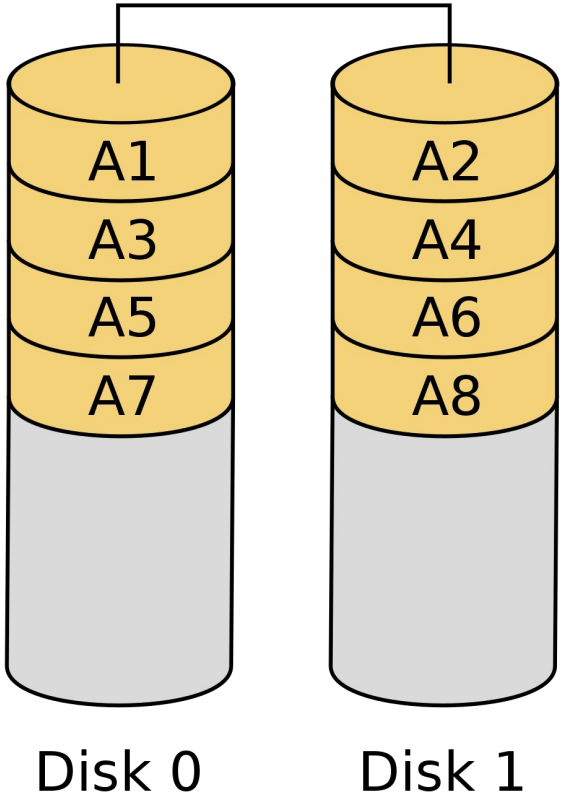


EPIC
Institute of Technology
Powered by epam

RAID

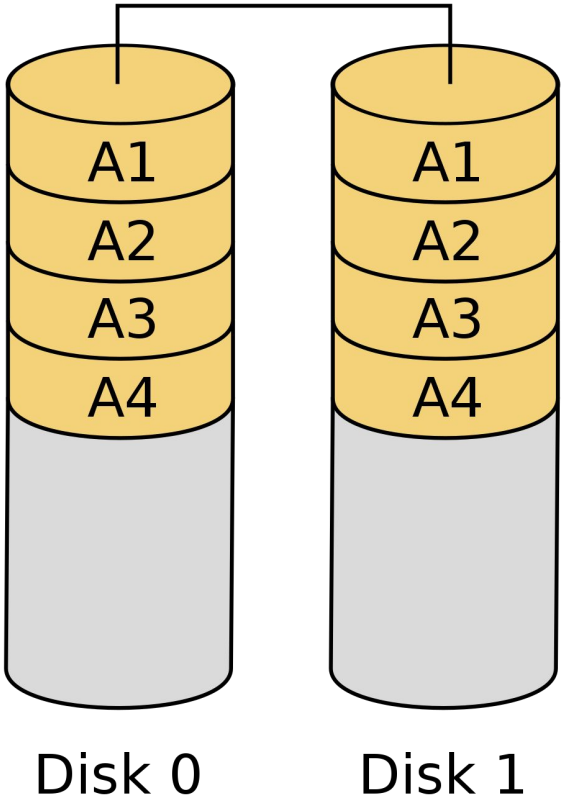
EPIC
Institute of Technology
Powered by epam

RAID 0



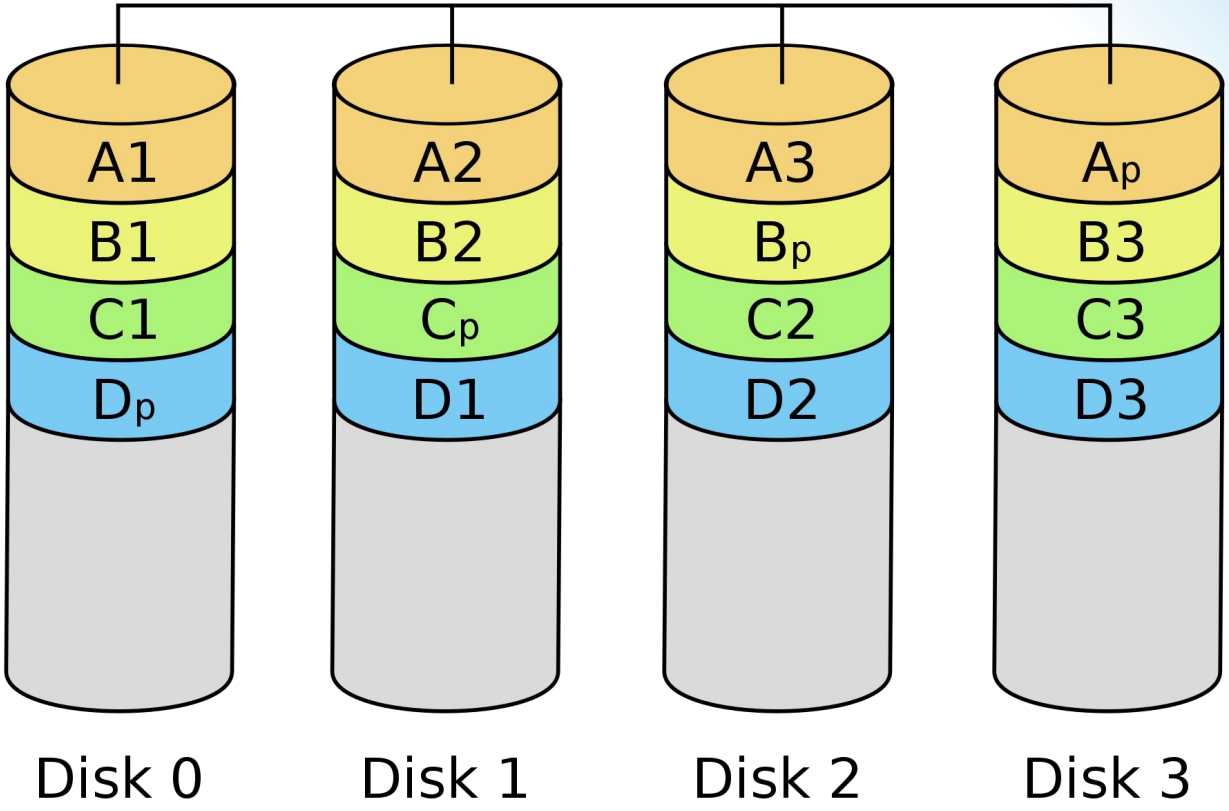
EPIC
Institute of Technology
Powered by epam

RAID 1



EPIC
Institute of Technology
Powered by epam

RAID 5



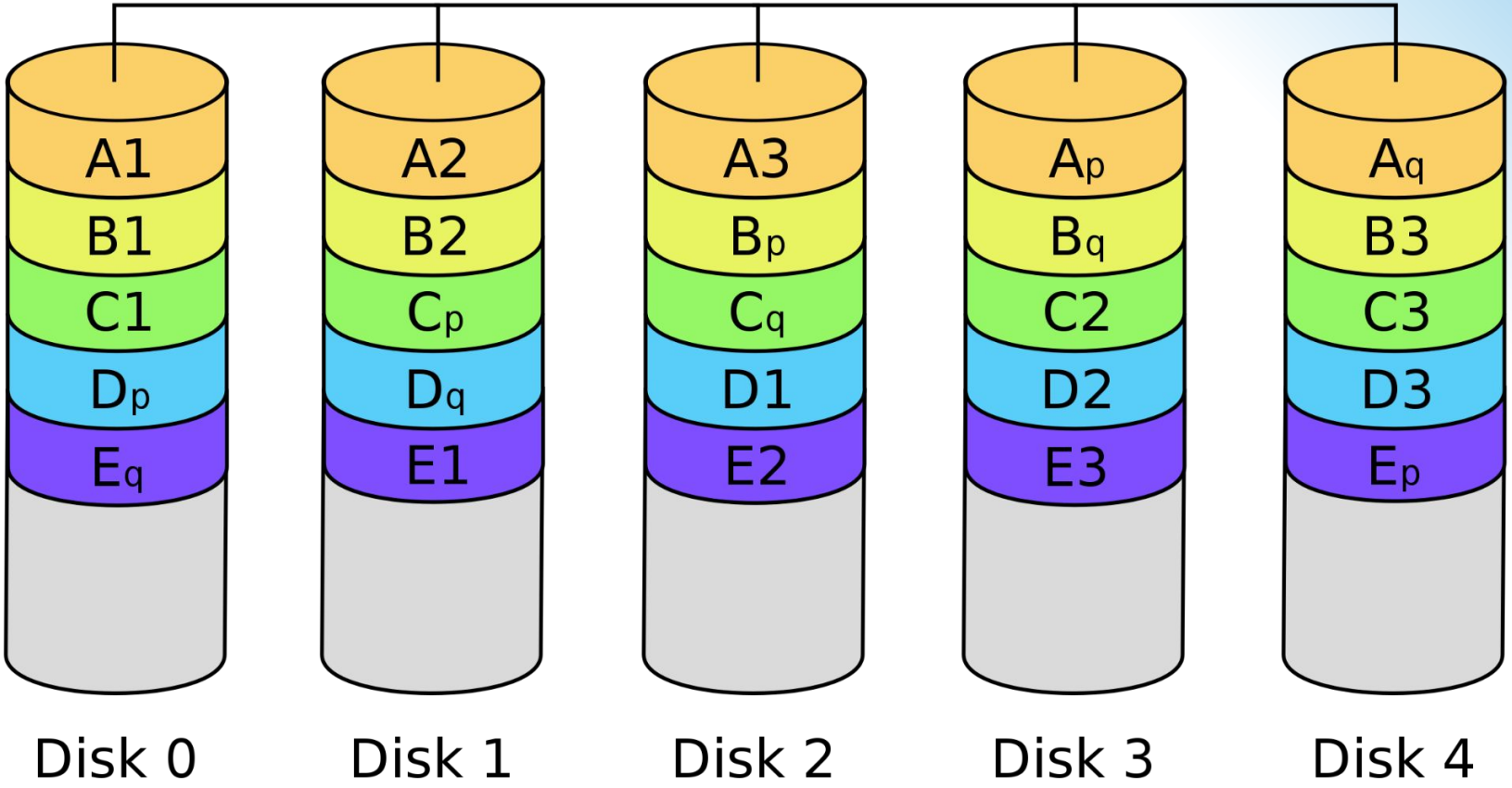
EPIC
Institute of Technology
Powered by epam

Paritet in RAID 5

1) A_1 XOR A_1 = 0 (always!!!!)
2) A_1 XOR A_2 XOR A_3 = num
3) num XOR A_1 XOR A_2 = A_3

EPIC
Institute of Technology
Powered by epam

RAID 6



EPIC Institute of Technology Powered by epam

Paritet in RAID 6

$$\mathbf{P} = \bigoplus_i \mathbf{D}_i = \mathbf{D}_0 \oplus \mathbf{D}_1 \oplus \mathbf{D}_2 \oplus \ldots \oplus \mathbf{D}_{n-1}$$

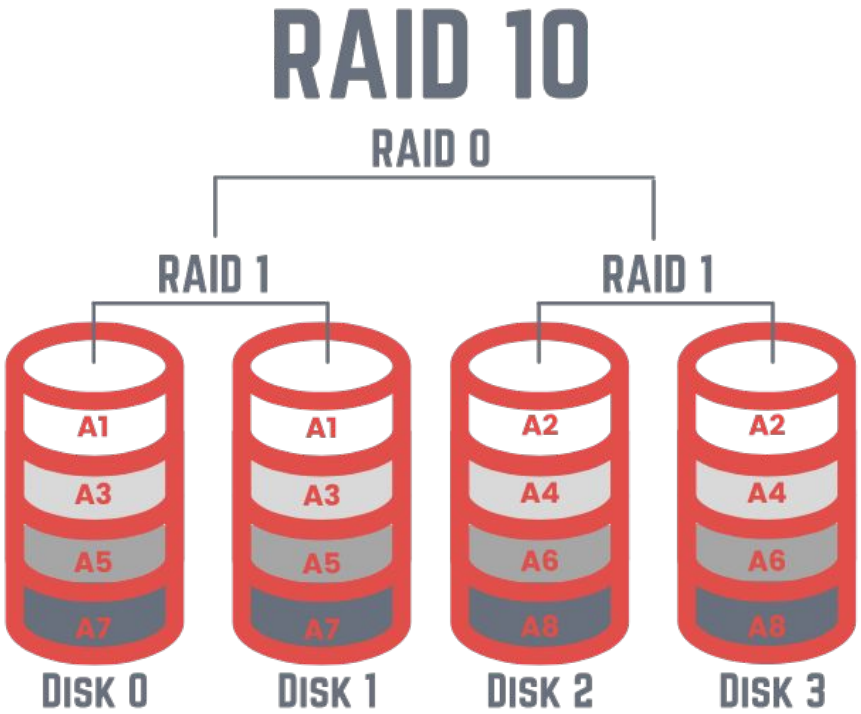
$$\mathbf{Q} = \bigoplus_i g^i \mathbf{D}_i = g^0\mathbf{D}_0 \oplus g^1\mathbf{D}_1 \oplus g^2\mathbf{D}_2 \oplus \ldots \oplus g^{n-1}\mathbf{D}_{n-1}$$

$$A = \mathbf{P} \oplus \left(\bigoplus_{\ell:\ \ell \neq i \text{ and } \ell \neq j} D_\ell \right) = D_i \oplus D_j$$

$$B = \mathbf{Q} \oplus \left(\bigoplus_{\ell:\ \ell \neq i \text{ and } \ell \neq j} g^\ell D_\ell \right) = g^i D_i \oplus g^j D_j$$

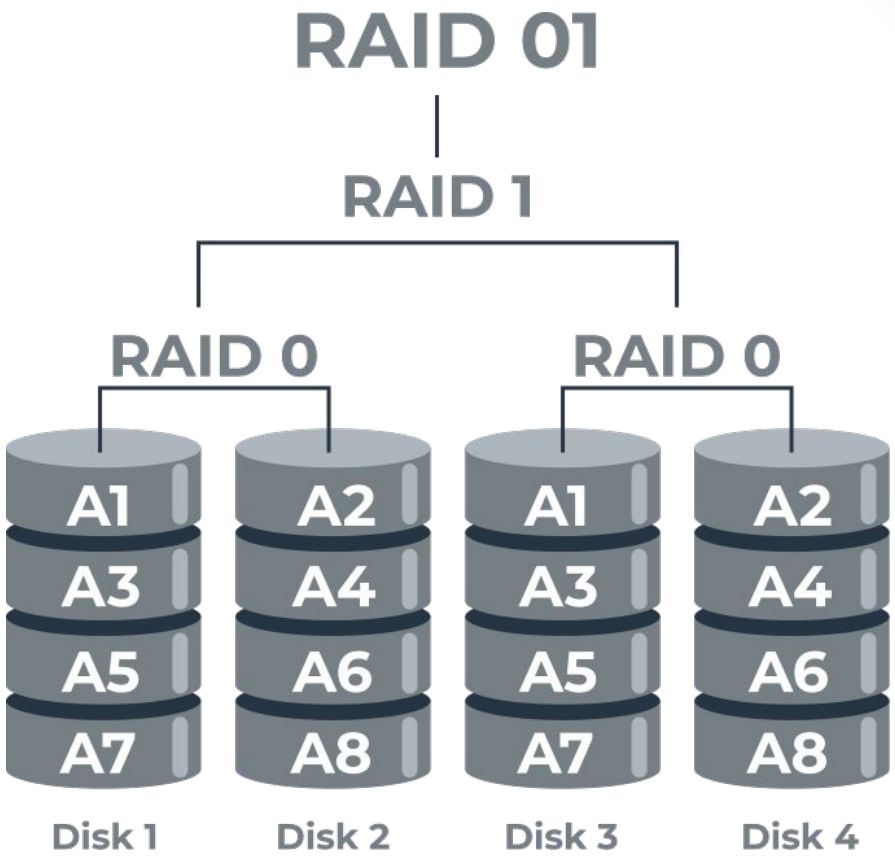
EPIC
Institute of Technology
Powered by epam

RAID 10 (RAID 1+0)



EPIC
Institute of Technology
Powered by <epam>

RAID 01 (RAID 0+1)



EPIC
Institute of Technology
Powered by epam

08

Cache-friendliness

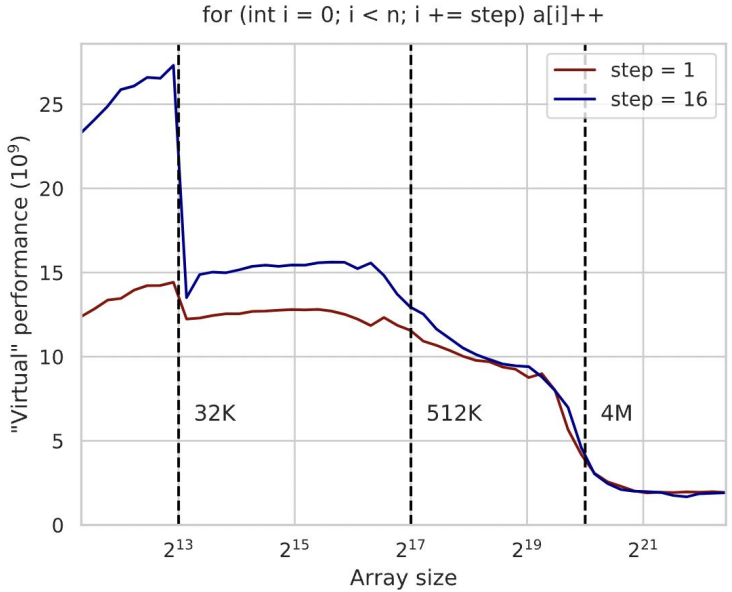
In this section, we will discuss Cache-friendly ideas.

EPIC Institute of Technology
Powered by <epam>

Funny thing.

```
for (int i = 0; i < N; i += D)
    a[i]++;
```

If we run it with $D = 1$ and $D = 16$, we can observe something interesting:



Performance is normalized by the total time to run benchmark, not the total number of elements incremented

EPIC
Institute of Technology
Powered by epam

Simple example

```
1  static void OneCycle() {
2      for (int i = 0; i < 10000; i++) {
3          z[i] = x[i] - y[i];
4          z[i] = z[i] * z[i];
5      }
6  }
7
8  static void TwoCycles() {
9      for (int i = 0; i < 10000; i++) {
10         z[i] = x[i] - y[i];
11     }
12     for (int i = 0; i < 10000; i++) {
13         z[i] = z[i] * z[i];
14     }
15 }
```

EPIC
Institute of Technology
Powered by <epam>

Comparison

Quick C++ Benchmark

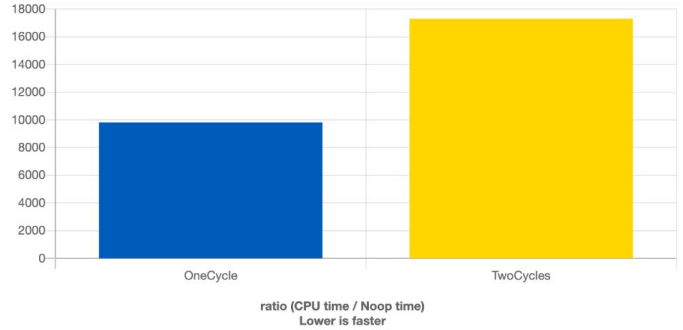
Run Quick Bench locally | Support Quick Bench Suite | More

```
  }
  for (auto _ : state) {
    for (int i = 0; i < 10000; i++) {
      z[i] = x[i] - y[i];
      z[i] = z[i] * z[i];
    }
  }
  int sum = 0;
  for (int i = 0; i < 10000; i++) {
    sum += z[i];
  }
  if (sum == 0) {
    exit(1);
  }
}

BENCHMARK(OneCycle);

static void TwoCycles(benchmark::State& state) {
  int z[10000], x[10000], y[10000];
  for (int i = 0; i < 10000; i++) {
    x[i] = i % 100;
    y[i] = i % 151;
  }
  for (auto _ : state) {
    for (int i = 0; i < 10000; i++) {
      z[i] = x[i] - y[i];
    }
    for (int i = 0; i < 10000; i++) {
      z[i] = z[i] * z[i];
    }
  }
  int sum = 0;
  for (int i = 0; i < 10000; i++) {
    sum += z[i];
  }
  if (sum == 0) {
    exit(1);
  }
}
BENCHMARK(TwoCycles);
```

compiler = Clang 15.0 | std = c++20 | optim = O2 | STL = libstdc++(GNU)

Run Benchmark | Record disassembly | Clear cached results

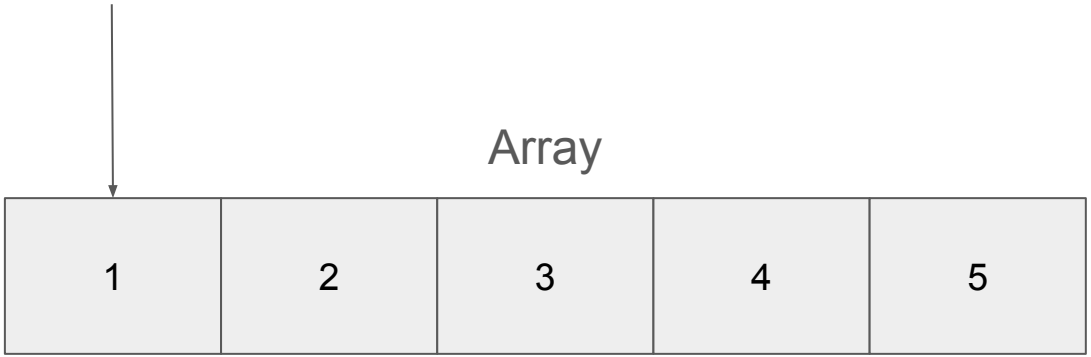
Charts | Assembly

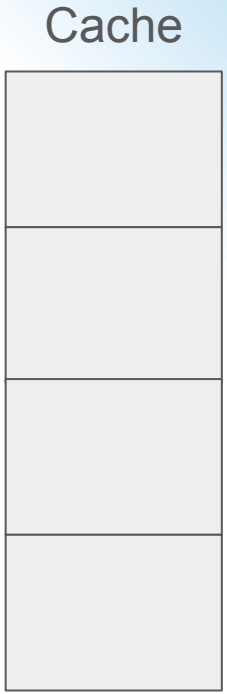


Show Noop bar

EPIC Institute of Technology
Powered by epam

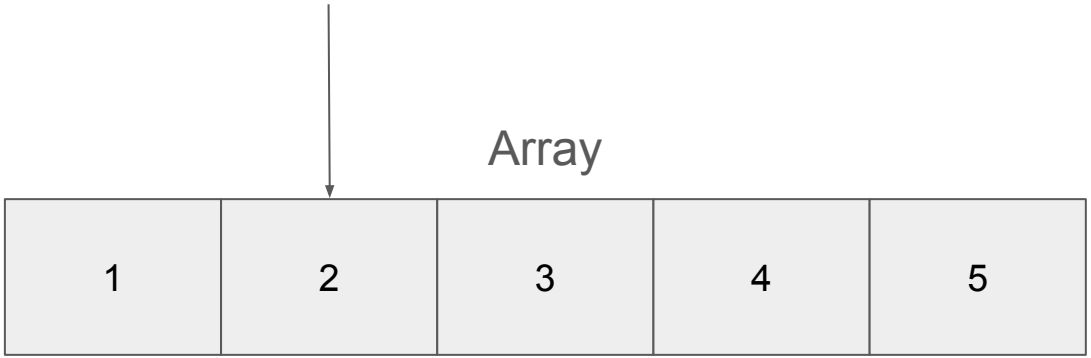
How is it working?

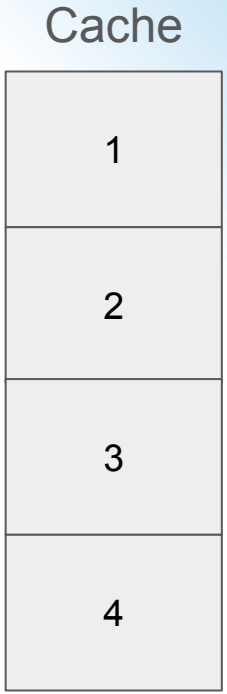




EPIC Institute of Technology
Powered by <epam>

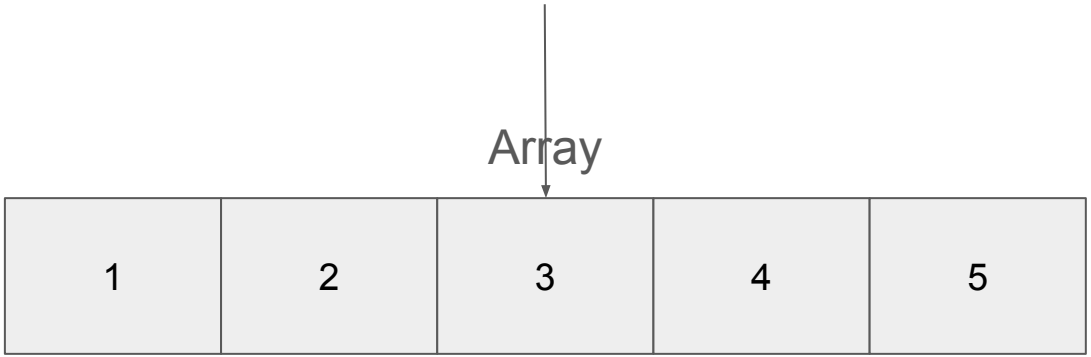
How is it working?

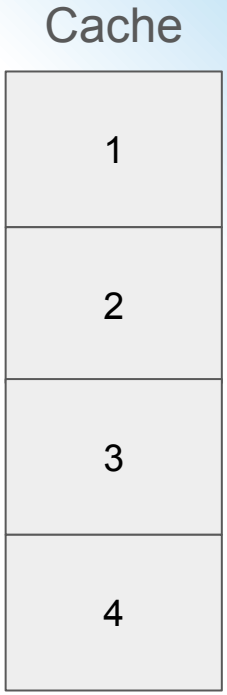




EPIC Institute of Technology
Powered by epam

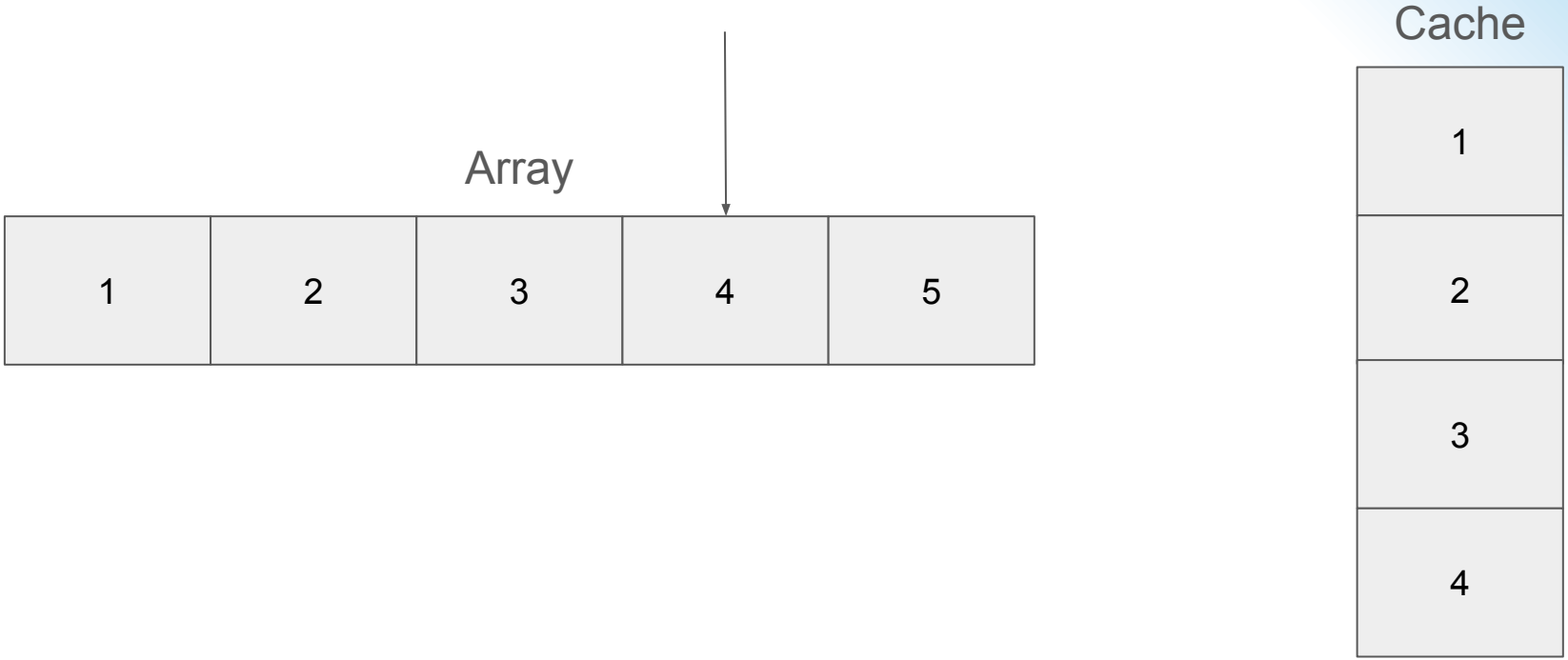
How is it working?





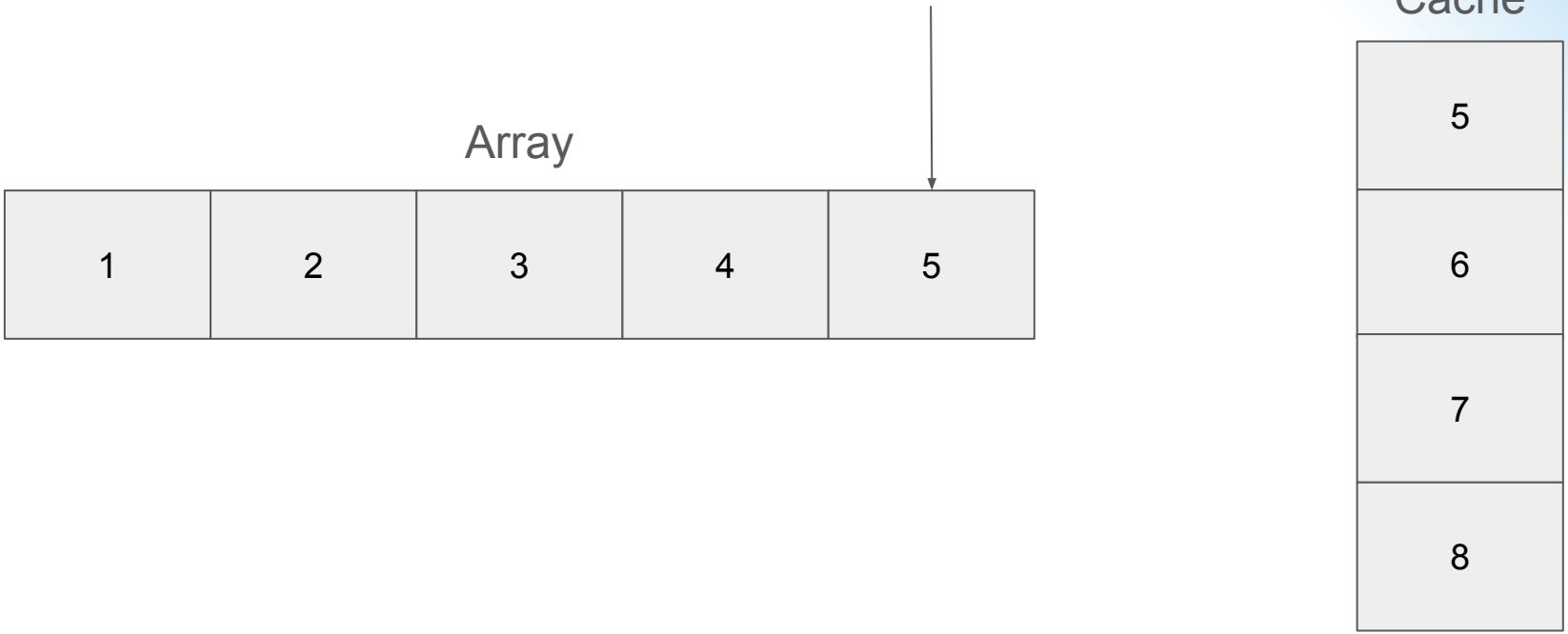
EPIC Institute of Technology
Powered by <epam>

How is it working?



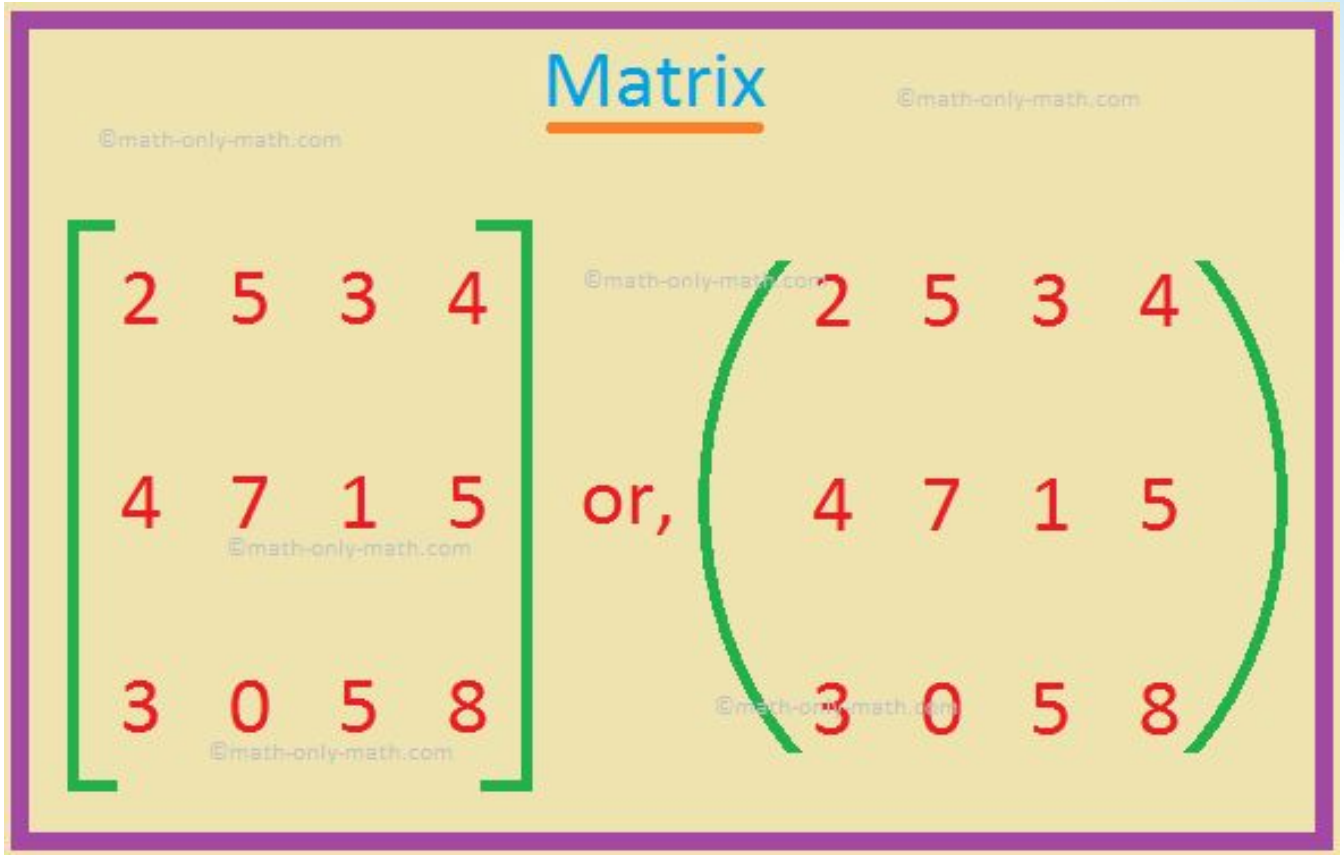
EPIC Institute of Technology
Powered by epam

How is it working?



EPIC
Institute of Technology
Powered by epam

Matrix



EPIC Institute of Technology
Powered by <epam>

Transpose of a matrix

$$A = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}_{2 \times 3} \qquad A^T = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}_{3 \times 2}$$

EPIC
Institute of Technology
Powered by <epam>

Transpose of a matrix

```cpp
int main() {
    int n, m;
    cin >> n >> m;
    int matrix[n][m], matrix_t[m][n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> matrix[i][j];
        }
    }

    // transpose
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            matrix_t[j][i] = matrix[i][j];
        }
    }

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            cout << matrix_t[i][j] << " ";
        }
        cout << "\n";
    }

    return 0;
}
```

Success #stdin #stdout 0.01s 5300KB

stdin

```
3 5
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
```

stdout

```
1 6 11
2 7 12
3 8 13
4 9 14
5 10 15
```

code

EPIC Institute of Technology
Powered by epam

Cache-friendly

```
15.     // transpose
16.     int block_size = 2;
17.
18.     for (int i = 0; i < n; i += block_size) {
19.         for (int j = 0; j < n; j += block_size) {
20.             for (int ii = i; ii < i + block_size && ii < n; ii++) {
21.                 for (int jj = j; jj < j + block_size && jj < n; jj++) {
22.                     matrix_t[jj][ii] = matrix[ii][jj];
23.                 }
24.             }
25.         }
26.     }
27.
28.     for (int i = 0; i < m; i++) {
29.         for (int j = 0; j < n; j++) {
30.             cout << matrix_t[i][j] << " ";
31.         }
32.         cout << "\n";
33.     }
34.
35.     return 0;
36. }
```

Success #stdin #stdout 0s 5296KB

stdin

```
4 4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

stdout

```
1 5 9 13
2 6 10 14
3 7 11 15
4 8 12 16
```

code

EPIC Institute of Technology
Powered by <epam>

Comparing

edit fork download copy

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
    int n = 10000, m = 10000;
    int matrix[n][m], matrix_t[m][n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            matrix[i][j] = (i + j);
        }
    }

    // transpose
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            matrix_t[j][i] = matrix[i][j];
        }
    }

    for (int i = 0; i < m; i += 100) {
        cout << matrix_t[i][i] << " ";
    }

    return 0;
}
```

Success #stdin #stdout 0.47s 784800KB comments ()

stdin copy

```
Standard input is empty
```

stdout copy

```
0 200 400 600 800 1000 1200 1400 1600 1800 2000 2200 2400 2600 2800 3000 3200 3400 3600 3800 4000
4200 4400 4600 4800 5000 5200 5400 5600 5800 6000 6200 6400 6600 6800 7000 7200 7400 7600 7800 80
00 8200 8400 8600 8800 9000 9200 9400 9600 9800 10000 10200 10400 10600 10800 11000 11200 11400 1
1600 11800 12000 12200 12400 12600 12800 13000 13200 13400 13600 13800 14000 14200 14400 14600 14
800 15000 15200 15400 15600 15800 16000 16200 16400 16600 16800 17000 17200 17400 17600 17800 180
00 18200 18400 18600 18800 19000 19200 19400 19600 19800
```

https://ideone.com/1JnNEr

```cpp
    int block_size = 4;

    for (int i = 0; i < n; i += block_size) {
        for (int j = 0; j < n; j += block_size) {
            for (int ii = i; ii < i + block_size && ii < n; ii++) {
                for (int jj = j; jj < j + block_size && jj < n; jj++) {
                    matrix_t[jj][ii] = matrix[ii][jj];
                }
            }
        }
    }

    for (int i = 0; i < m; i += 100) {
        cout << matrix_t[i][i] << " ";
    }

    return 0;
}
```

Success #stdin #stdout 0.34s 784732KB comments (?)

stdin copy

```
Standard input is empty
```

stdout copy

```
0 200 400 600 800 1000 1200 1400 1600 1800 2000 2200 2400 2600 2800 3000 3200 3400 3600 3800 4000
4200 4400 4600 4800 5000 5200 5400 5600 5800 6000 6200 6400 6600 6800 7000 7200 7400 7600 7800 80
00 8200 8400 8600 8800 9000 9200 9400 9600 9800 10000 10200 10400 10600 10800 11000 11200 11400 1
1600 11800 12000 12200 12400 12600 12800 13000 13200 13400 13600 13800 14000 14200 14400 14600 14
800 15000 15200 15400 15600 15800 16000 16200 16400 16600 16800 17000 17200 17400 17600 17800 180
00 18200 18400 18600 18800 19000 19200 19400 19600 19800
```

https://ideone.com/t3jlbT

EPIC
Institute of Technology
Powered by epam

Cache misses

Miss	Hit	Hit	Hit	Miss	Hit	Hit	Hit	Miss	Hit
Miss	Hit	Hit	Hit	Miss	Hit	Hit	Hit	Miss	Hit
Miss	Hit	Hit	Hit	Miss	Hit	Hit	Hit	Miss	Hit

EPIC Institute of Technology
Powered by <epam>

```cpp
int M[size][size];

int matrixsum1() {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            sum += M[i][j];
        }
    }

    return sum % 1503;
}

int matrixsum2() {
    int sum = 0;
    for (int i = 0; i < size; i++){
        for (int j = 0; j < size; j++) {
            sum += M[j][i];
        }
    }
    return sum % 1503;
}

int main() {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            M[i][j] = (i + j) % 100;
        }
    }
    int ans = 0;
    for (int i = 0; i <= size; i++) {
        ans += matrixsum1();
    }
    cout << ans;
    return 0;
}
```

https://ideone.com/TJrI23

```
30. int main() {
31.     for (int i = 0; i < size; i++) {
32.         for (int j = 0; j < size; j++) {
33.             M[i][j] = (i + j) % 100;
34.         }
35.     }
36.     int ans = 0;
37.     for (int i = 0; i <= size; i++) {
38.         ans += matrixsum1();
39.     }
40.     cout << ans;
41.     return 0;
42. }
```

Success #stdin #stdout 0.5s 7504KB

stdin

```
Standard input is empty
```

stdout

```
198198
```

```
30. int main() {
31.     for (int i = 0; i < size; i++) {
32.         for (int j = 0; j < size; j++) {
33.             M[i][j] = (i + j) % 100;
34.         }
35.     }
36.     int ans = 0;
37.     for (int i = 0; i <= size; i++) {
38.         ans += matrixsum2();
39.     }
40.     cout << ans;
41.     return 0;
42. }
```

Success #stdin #stdout 1.33s 7456KB

stdin

```
Standard input is empty
```

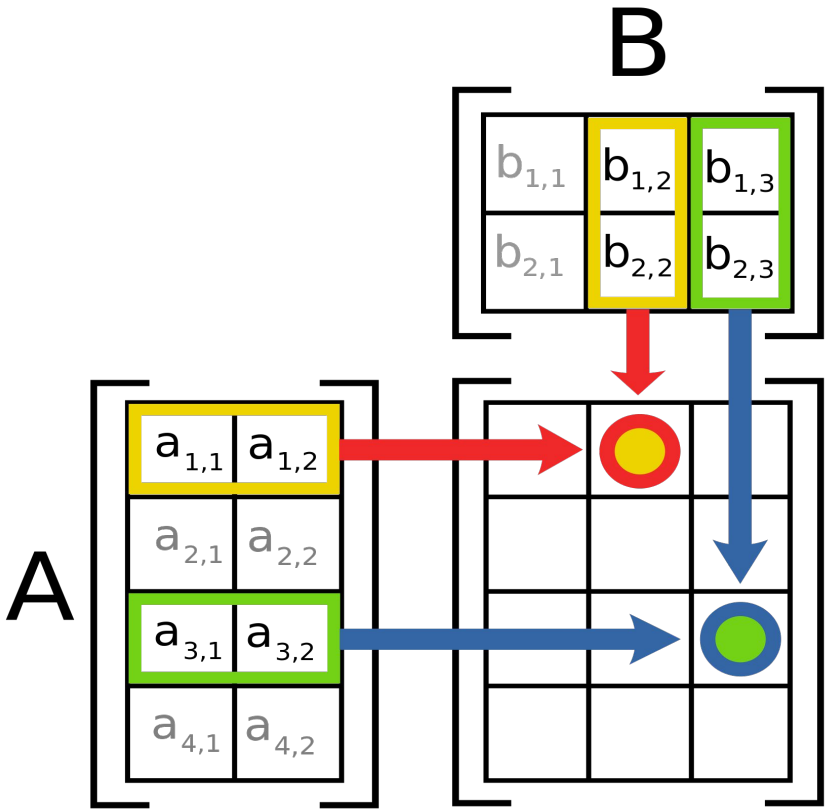
stdout

```
198198
```

https://ideone.com/1S72vq

EPIC
Institute of Technology
Powered by epam

Multiplication



EPIC Institute of Technology
Powered by epam

Multiplication

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

A B C

A, B and C are square metrices of size N x N
a, b, c and d are submatrices of A, of size N/2 x N/2
e, f, g and h are submatrices of B, of size N/2 x N/2

EPIC Institute of Technology
Powered by epam

Multiplication

```
static void MatrixMultIJK() {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            int sum = 0;
            for (int k = 0; k < n; k++) {
                sum += A[i][k] * B[k][j];
            }
            C[i][j] = sum;
        }
    }
}
```

```
78.  int main() {
79.      for (int i = 0; i < n; i++) {
80.          for (int j = 0; j < n; j++) {
81.              A[i][j] = (i + j) % 5;
82.              B[i][j] = (i + j) % 5;
83.          }
84.      }
85.      MatrixMultIJK();
86.      cout << "A : \n";
87.      for (int i = 0; i < n; i++) {
88.          for (int j = 0; j < n; j++) {
89.              cout << A[i][j] << " ";
90.          }
91.          cout << "\n";
92.      }
93.      cout << "B : \n";
94.      for (int i = 0; i < n; i++) {
95.          for (int j = 0; j < n; j++) {
96.              cout << B[i][j] << " ";
97.          }
98.          cout << "\n";
99.      }
100.     cout << "C : \n";
101.     for (int i = 0; i < n; i++) {
102.         for (int j = 0; j < n; j++) {
103.             cout << C[i][j] << " ";
104.         }
105.         cout << "\n";
106.     }
107. }
```

stdout

```
A :
0 1 2 3 4
1 2 3 4 0
2 3 4 0 1
3 4 0 1 2
4 0 1 2 3
B :
0 1 2 3 4
1 2 3 4 0
2 3 4 0 1
3 4 0 1 2
4 0 1 2 3
C :
30 20 15 15 20
20 30 20 15 15
15 20 30 20 15
15 15 20 30 20
20 15 15 20 30
```

```
static void MatrixMultIJK() {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            int sum = 0;
            for (int k = 0; k < n; k++) {
                sum += A[i][k] * B[k][j];
            }
            C[i][j] = sum;
        }
    }
}
```

```
static void MatrixMultJIK() {
    for (int j = 0; j < n; j++) {
        for (int i = 0; i < n; i++) {
            int sum = 0;
            for (int k = 0; k < n; k++) {
                sum += A[i][k] * B[k][j];
            }
            C[i][j] = sum;
        }
    }
}
```

```
static void MatrixMultKIJ() {
    for (int k = 0; k < n; k++) {
        for (int i = 0; i < n; i++) {
            int r = A[i][k];
            for (int j = 0; j < n; j++) {
                C[i][j] += r * B[k][j];
            }
        }
    }
}
```

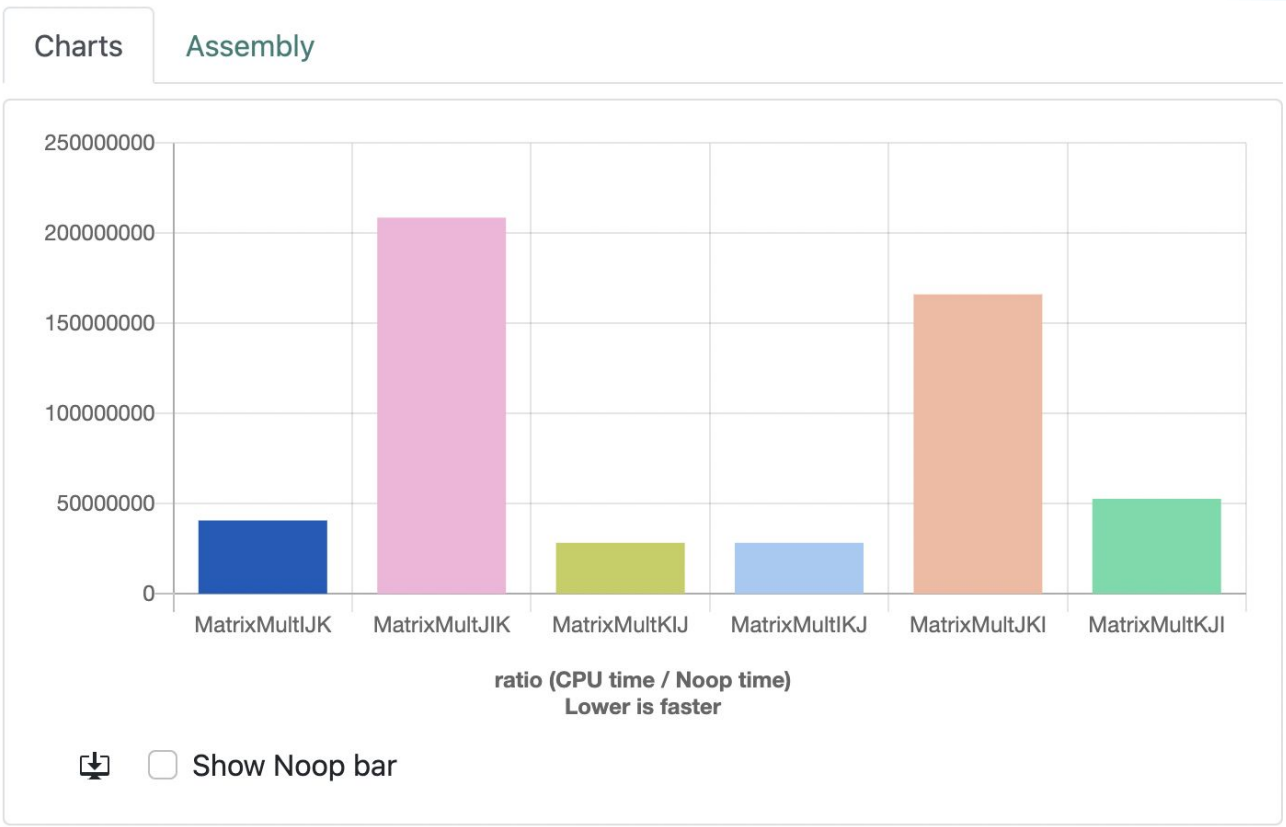
```
static void MatrixMultIKJ() {
    for (int i = 0; i < n; i++) {
        for (int k = 0; k < n; k++) {
            int r = A[i][k];
            for (int j = 0; j < n; j++) {
                C[i][j] += r * B[k][j];
            }
        }
    }
}
```

```
static void MatrixMultJKI() {
    for (int j = 0; j < n; j++) {
        for (int k = 0; k < n; k++) {
            int r = B[k][j];
            for (int i = 0; i < n; i++) {
                C[i][j] += A[i][k] * r;
            }
        }
    }
}
```

```
static void MatrixMultKJI() {
    for (int k = 0; k < n; k++) {
        for (int j = 0; j < n; j++) {
            int r = B[k][j];
            for (int i = 0; i < n; i++) {
                C[i][j] += A[i][k] * r;
            }
        }
    }
}
```

EPIC
Institute of Technology
Powered by <epam>

Comparison

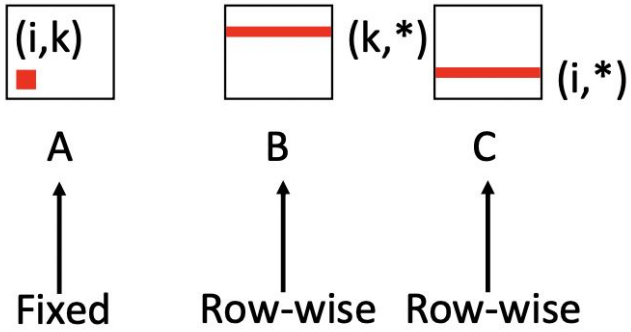


EPIC Institute of Technology
Powered by epam

Why?

```
for (k=0; k<n; k++) {
  for (i=0; i<n; i++) {
    r = A[i][k];
    for (j=0; j<n; j++)
      C[i][j] += r * B[k][j];
  }
}
```

Inner loop:



Misses per inner loop iteration:

A	B	C
0.0	0.25	0.25

EPIC
Institute of Technology
Powered by epam

That's All Folks!