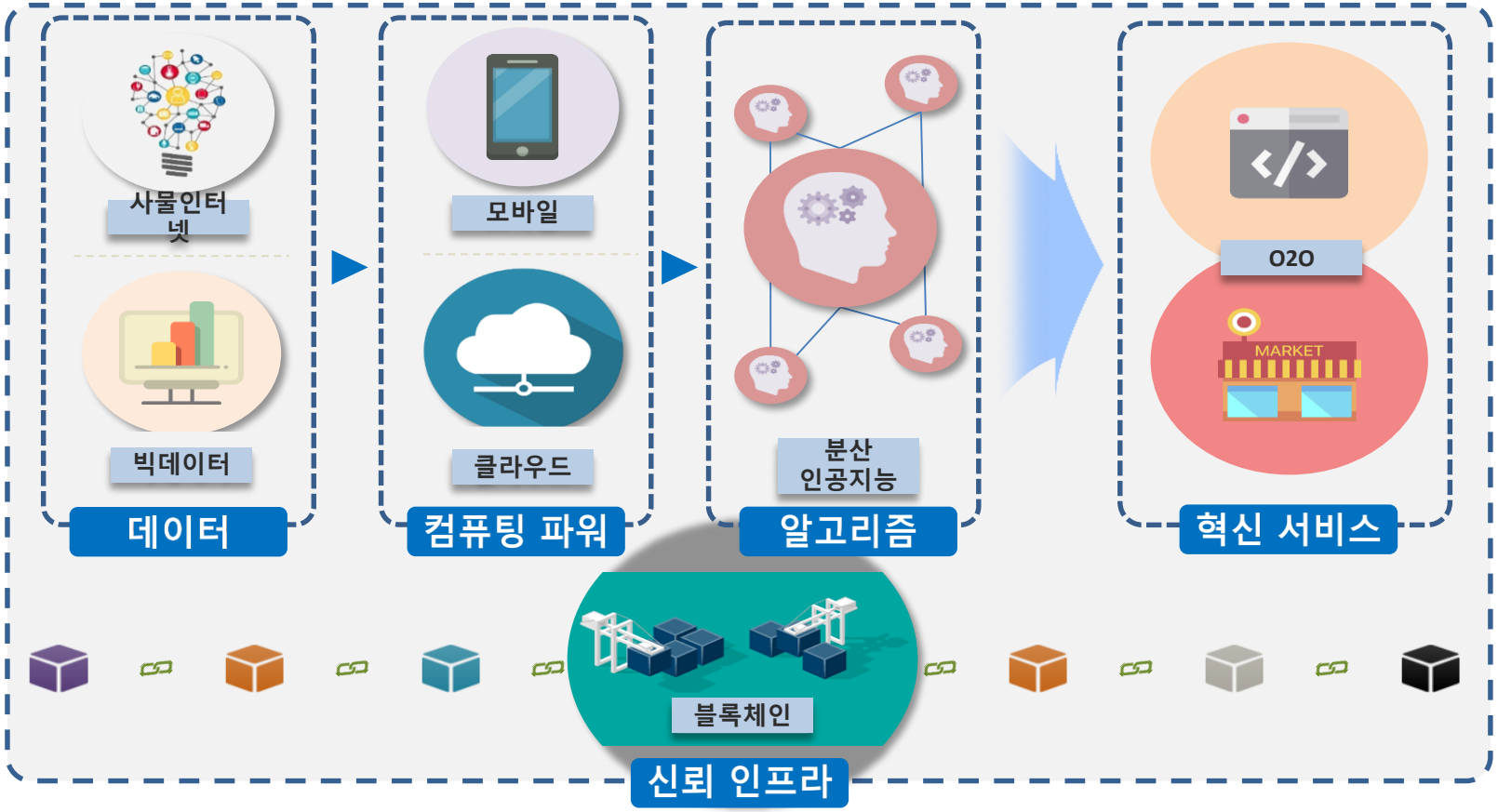
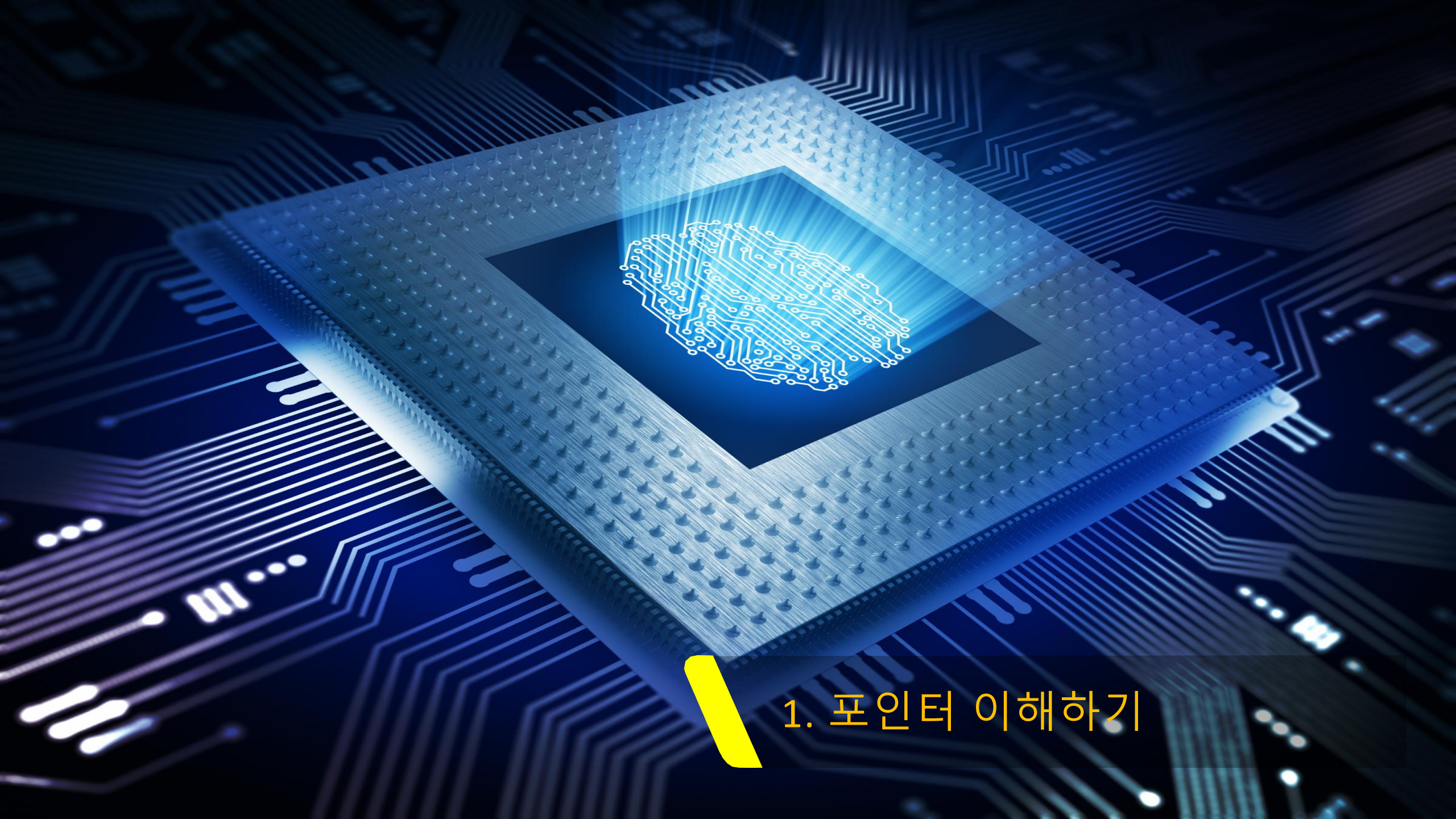


Hanyang Univ.

자료구조론 3주차

강 진 영

CONTENTS



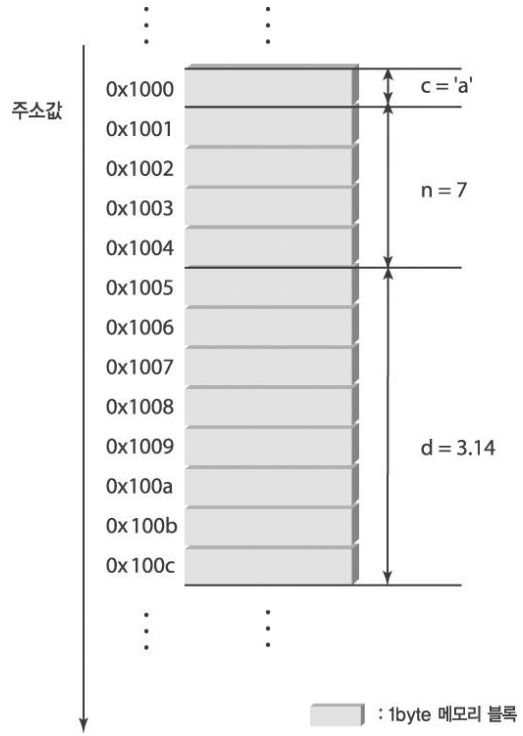
1. 포인터 이해하기

Contents

포인터란

- **포인터와 포인터 변수**
  - 메모리의 주소 값을 저장하기 위한 변수
  - “포인터 ” 를 흔히 “포인터 변수 ”
  - 주소 값과 포인터는 상이

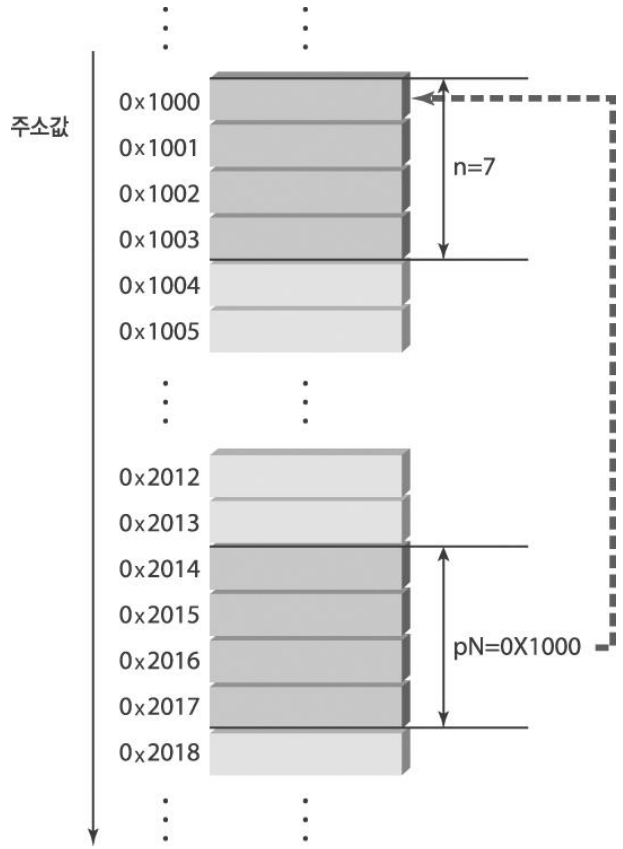
```
int main(void)
{
    char c='a';
    int n=7;
    double d=3.14;
    . . . . .
```



Contents

포인터란

- 그림을 통한 포인터의 이해
  - 컴퓨터의 주소 체계에 따라 크기가 결정
  - 32비트 시스템 기반 : 4 바이트



Contents

포인터란

- 포인터의 타입과 선언
  - 포인터 선언 시 사용되는 연산자 : *
  - A형 포인터(A*) : A형 변수의 주소 값을 저장

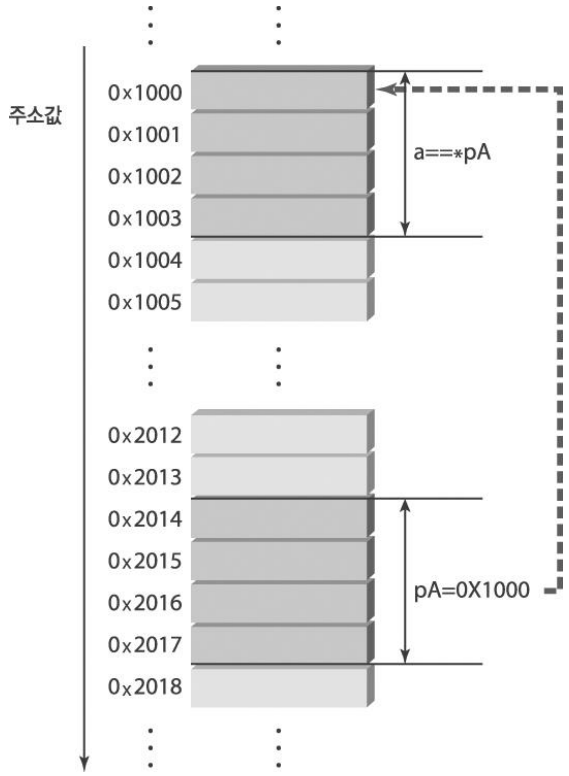
```
int main(void)
{
   int *a;              // a라는 이름의 int형 포인터
   char *b;             // b라는 이름의 char형 포인터
   double *c;// c라는 이름의 double형 포인터
    . . . . .
```

Contents

포인터란

- 주소 관련 연산자
  - ➢ & 연산자 : 변수의 주소 값 반환
  - ➢ * 연산자 : 포인터가 가리키는 메모리 참조

```
int main(void)
{
   int a=2005;
   int *pA=&a;
   printf("%d", a);   //직접 접근
   printf("%d", *pA);   // 간접 접근
   . . . . .
```



Contents

포인터란

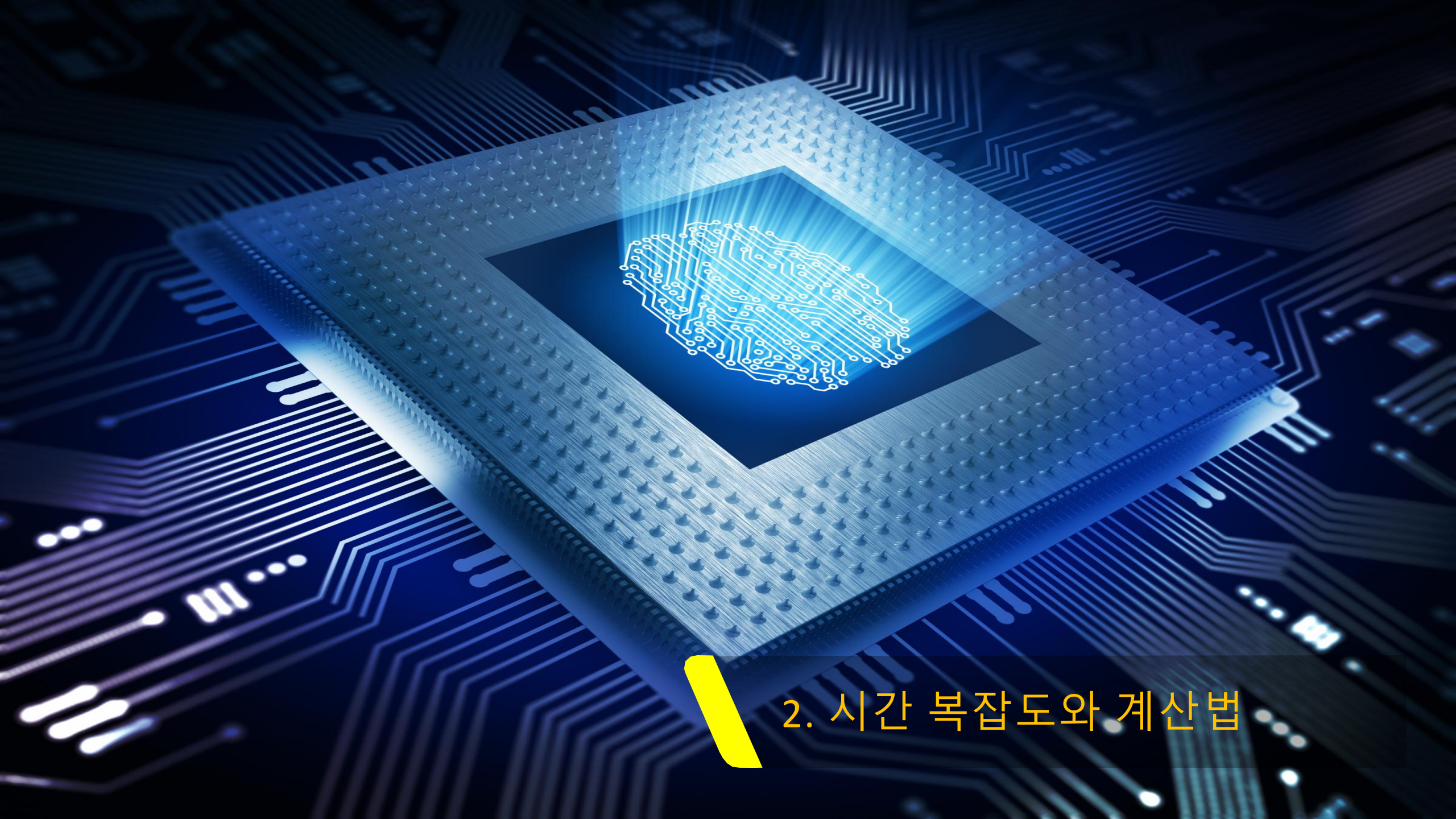
```
#include <stdio.h>

int main(void)
{
    int a=2005;
    int* pA=&a;

    printf("pA : %d ₩n", pA);
    printf("&a : %d ₩n", &a);

    (*pA)++;          //a++와 같은 의미를 지닌다.

    printf("a   : %d ₩n", a);
    printf("*pA : %d ₩n", *pA);

    return 0;
}
```

Contents

포인터란

- 포인터에 다양한 타입이 존재하는 이유
  - ➢ 포인터 타입은 참조할 메모리의 크기 정보를 제공

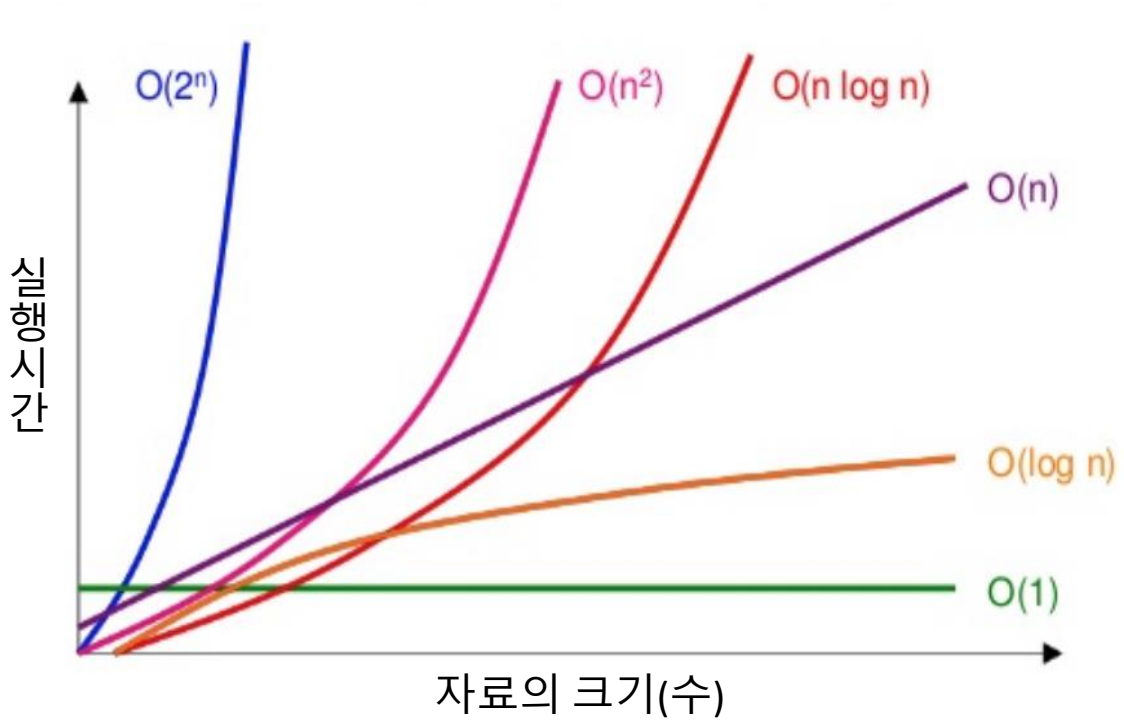
```
#include <stdio.h>
 int main(void)
 {
     int a=10;
     int *pA = &a;
     double e=3.14;
     double *pE=&e;

     printf("%d %f", *pA, *pE);
     return 0;
 }
```



2. 시간 복잡도와 계산법

시간 복잡도 – 연산횟수의 총량

- 1(상수)
  - 입력 자료의 크기와 관계없이 일정한시간
  - 해시 검색 알고리즘
- log N
  - Divide & Conquer
  - 이진검색, 이진트리검색
- N
  - Scan
  - 선형검색
- N log N
  - Divide & Conquer 또는 Merge
  - 병합정렬
- N^2
  - 이중 반복



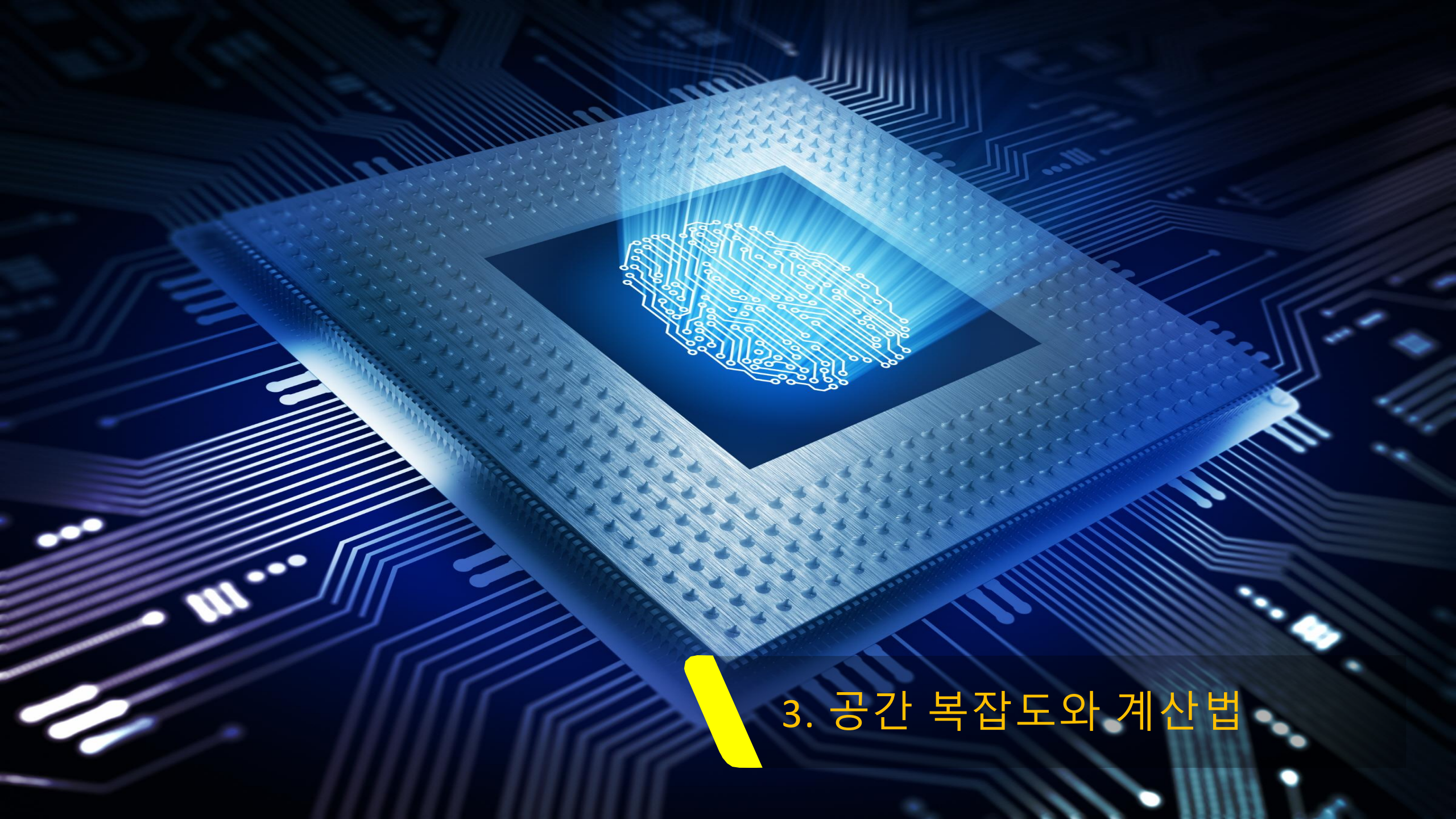
- $O(1) < O(\log n) < O(n) < O(n\log n) < O(n^2) < O(n^3)$

시간 복잡도 계산법

➢ 각 코드의 시간 복잡도는?

```
#include <stdio.h>

void main(void)
{
    int tmpNum1 = 10;
    int tmpNum2 = 20;
    int tmpNum3 = 30;

    for(tmpNum1; tmpNum1<tmpNum3; tmpNum1++){
       for(for(tmpNum1; tmpNum1<tmpNum3; tmpNum1++){
          int x = tmpNum1 * tmpNum3;
          int y = tmpNum1 * tmpNum1;
          int z = tmpNum3 * tmpNum3;
       }
    }

    for(tmpNum2; tmpNum2<tmpNum3; tmpNum2++){
       int tmpNum4 = tmpNum2 * tmpNum3;
    }

    int tmpNum5 = tmpNum3 * tmpNum4;
}
```

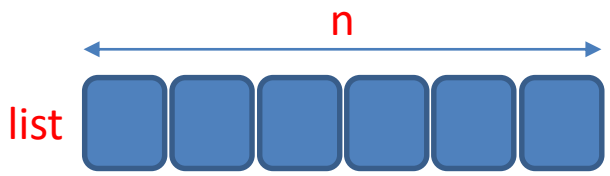
```
Def findMin(alist) :
   minsofar = alist[0]
   for i in alist :
      if i < minsofar :
         minsofar = i
   return minsofar

print(findMin([5, 2, 1, 0]))
print(findMin([0, 2, 3, 4]))

for listSize in range(1000, 1001, 1000) :
   alist = [randrange(100000) for _ in range(listSize)]
   start = time.time()
   print(findMin(alist))
   end = time.time()
   print("size : {}, time: {}".format(listSize, end-start))
```



3. 공간 복잡도와 계산법

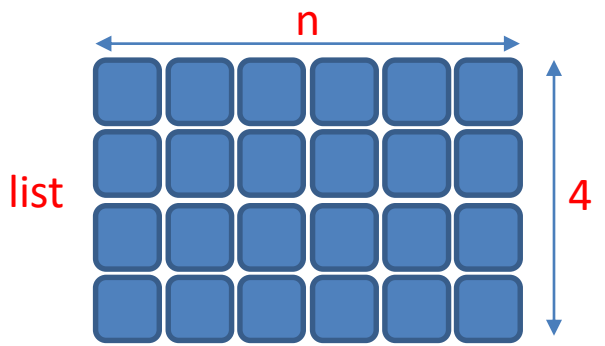
공간 복잡도 – 가용되는 메모리의 총량

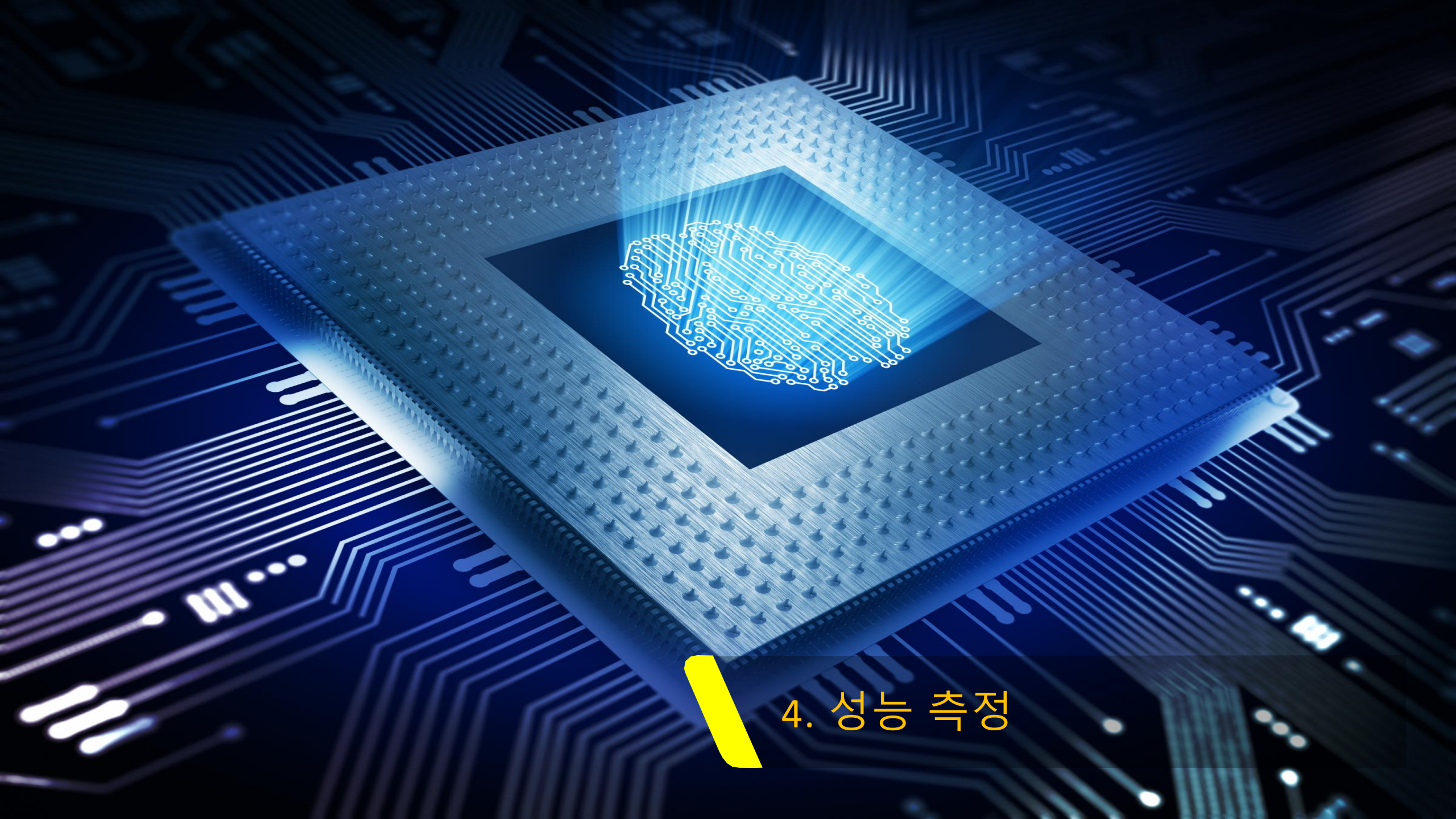
```
#include <stdio.h>

float sum(float list[], int n)
{
   float tempSum = 0.0f;
   int i = 0;

   for(i=0; i<n; i++){
      tempSum += list[i];
   }

   return tempSum;
}
```



list $(a_{11}, a_{12}, a_{13}, \cdots, a_{1n})$

```
#include <stdio.h>

float sum(float list[][4], int n)
{
   float tempSum = 0.0f;
   int i = 0, j = 0;

   for(i=0; i<n; i++){
      for(j=0; j<4; j++){
         tempSum += list[i];
      }
   }
   return tempSum;
}
```





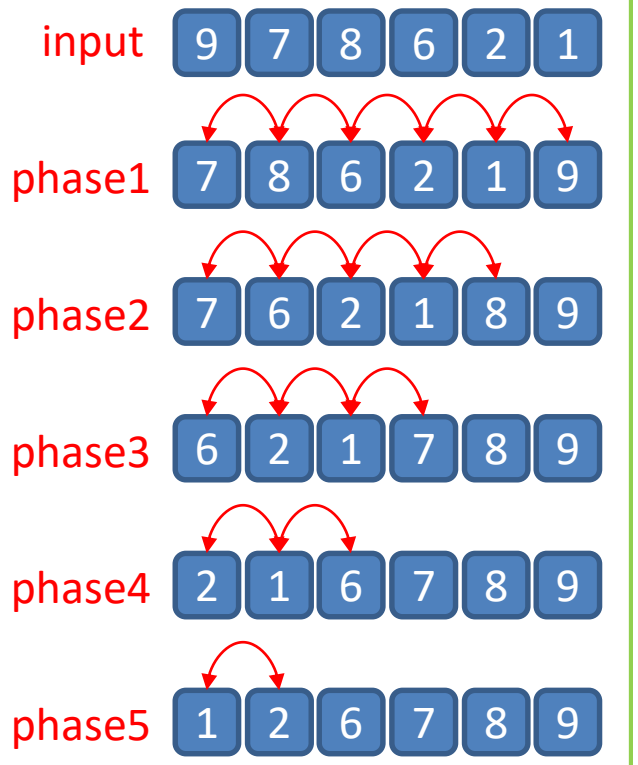
4. 성능 측정

Contents

성능 측정 – 버블정렬

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define ATOM_SIZE 10000

int main(int argc, char* argv[], char* env[])
{
   srand(time(NULL));
   clock_t start = 0.0f, end = 0.0f;
   int atomCarrier[ATOM_SIZE] = {0,};
   int idx = 0, idy = 0;

   Shuffle(atomCarrier);

   start = clock();

   for(; idx<ATOM_SIZE; idx++){
      for(idy=1; idy<ATOM_SIZE; idy++){
         if(atomCarrier[idy] < atomCarrier[idy-1]){
            Swap(&atomCarrier[idy], &atomCarrier[idy-1]);
         }
      }
   }

   end = clock();

   printf("초 : %0.4lf ₩n, 밀리초 : %0.4lf ₩n",
               (double)(end-start)/1000, (double)(end-start));
}
```

```
int Swap(int* headNum, int* tailNum)
{
   int temp = *headNum;
   *headNum = *tailNum;
   *tailNum = temp;
}

void Shuffle(int* container)
{
   int idx = 0;

   for(; idx<ATOM_SIZE; idx++){
      container[idx] = rand()%1000;
   }
}
```



Q & A