LAPORAN PRAKTIKUM

ORGANISASI DAN ARSITEKTUR KOMPUTER

PERCOBAAN 2

Nama : Akbar Dwi Herlambang

NIM : 2308979

Kelas : 2-C

- **Tugas Pendahuluan**

1. Dari yang telah saya pelajari, tipe data *float* disimpan dalam ukuran 4 byte dengan rentang nilai minimun 1.2E-38 dan maksimum 3.4E+38 sedangkan untuk tipe data *double* disimpan dalam ukuran 8 byte dengan rentang nilai minimum 2.2E-308 dan maksimum 1.8E+308.
2. Setelah saya pelajari, perbedaan diantara keduanya adalah :
   - Logical Right Shift, akan digunakan ketika fungsi dari variable adalah unsigned

```
unsigned int test = 10;
test = test >> 1;
```

   - Arithmetic Right Shift, akan digunakan ketika most significant bit, mengidentifikasi signed variable tersebut sesuai dengan aturan two's complement 0 dan 1.

```
int x = -10;
x = x >> 1;
```

3. **A.** Ukuran Tipe Data daftar_NA_1
   - **"char"** kelas : 1 byte
   - **"short"** kode_matakuliah : 2 byte
   - "**int"** nim : 4 byte
   - **"char"** nilai_abjad : 1 byte
   - **"int"** nilai_angka : 4 byte

   Total : 12 byte

   Ukuran Tipe Data daftar_NA_2

   - **"char"** kelas : 1 byte
   - **"char"** nilai_abjad : 1 byte
   - **"short"** kode_matakuliah : 2 byte
   - **"int"** nim : 4 byte
   - **"int"** nilai_angka : 4 byte

   Total : 12 byte

   Representasi Data Memori daftar_NA_1

   | Kelas | Kode_Mata Kuliah | NIM | Nilai Abjad | Nilai Angka |

   Representasi Data Memori daftar_NA_2

   | Kelas | Nilai Abjad | Kode Mata Kuliah | Nilai Abjad | Nilai Angka |

B. Dari apa yang saya pahami, meskipun memiliki total ukuran byte yang sama namun daftar_NA_1 memiliki tipe data yang diacak sehingga menyebabkan beban terhadap alignment byte dalam memori karena ukuran dari setiap tipe data tiap perpindahan/pengeksekusian instruksi berbeda, beda dengan daftar_NA_2 yang memiliki tipe data terstruktu sehingga alignment byte tidak terlalu terbebani.

4. Dimisalkan, variable “arr[0][0] = 1”, “arr[0][1] = 2”, “arr[1][0] = 3”, dan “arr[1][1] = 4” maka penggambaran pada memori akan menjadi :

   | 1 | 2 | | arr[0][0] | arr[0][1] |

   | 3 | 4 | | arr[1][0] | arr[1][1] |

5. > input_no_5.c

```c
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul          : 1
// Percobaan      : 2
// Tanggal        : 22 April 2024
    // Nama (NIM)     : Akbar Dwi Herlambang (2308979)
    // Nama File      : input_no_5.c
    // Deskripsi      : Program Input untuk no_5.c

    #include <stdio.h>
    #include "input_no_5.h"

    int input()
    {

        double num;
        printf("Masukkan bilangan : ");
        scanf("%d", &num);

        return num;
}
```

> no_5.c

```c
// Organisasi dan Arsitektur Komputer Tekom 2B
// Modul         : 1
// Percobaan     :     2
// Tanggal       : 22 April 2024
// Nama (NIM)    :     Akbar Dwi Herlambang (2308979)
// Nama File     :     no_5.c
// Deskripsi     :     Demonstrasi Pointer

#include <stdlib.h>
#include <stdio.h>
#include "no_5.h"

typedef unsigned char *byte_pointer;
// address  = alamat dari variabel dalam memory
// size = ukuran bariable dalam memory (sizeoff)

void printByte(byte_pointer address, int size)
{
    int i;
    for (i = size-1; i >= 0; i--)
    {
        printf(" %.2x", address[i]);
    }
    printf("\n");
}
```

```c
void printBit(size_t const size, void const * const address)
{
    unsigned char *b = (unsigned char*) address;
    unsigned char byte;

    int i, j; int space; space=0; printf(" ");
    for (i=size-1;i>=0;i--)
    {
        for (j=7;j>=0;j--)
    {

    byte = b[i] & (1<<j); byte >>= j; printf("%u", byte); space++;

    if (space>=4)
        {
            printf(" "); space=0;
        }
    }
}
    puts("");
}
```

> main_prog,c

```c
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : main_prog.c
// Deskripsi       : Program gabungan/kompilasi untuk input_no_5.c
dan no_5.c

#include <stdio.h>
#include "input_no_5.h"
#include "no_5.h"

typedef unsigned char *byte_pointer;

int main()
{
      int number;

      number = input();

      printf("Representasi dalam byte: ");
    printByte((byte_pointer)&number, sizeof(int));

    printf("Representasi dalam bit: ");
    printBit(sizeof(int), &number);

    return 0;
}
```

> rep_byte_bit.exe

```
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\NO 5>rep_byte_bit.exe
Masukkan bilangan : 2
Representasi dalam byte:  00 00 00 02
Representasi dalam bit:  0000 0000 0000 0000 0000 0000 0000 0010

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\NO 5>
```

- **Tugas 1**

1. Source Code

```c
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : bit-xor.c
// Deskripsi       : Membuat Program BIT XOR

#include <stdio.h>

int bitXor(int x, int y)
{
    int result = (~x & y);
    result |= (~y & x);
    return result;
}

int main()
{
    int result = bitXor(4, 5);
    printf("Result: %d\n", result);
    return 0;
}
```

2. Hasil Eksekusi

```
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 1>gcc -o bit-xor.exe bit-xor.c

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 1>bit-xor.exe
Result: 1

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 1>bit-xor.exe
Result: 1
```

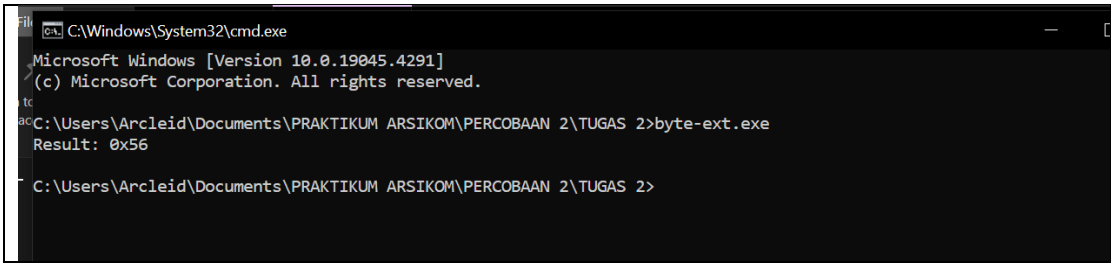
- **Tugas 2 : Fungsi Ekstraksi Byte**

1. Source Code

```c
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : byte-ext.c
// Deskripsi       : Membuat Program Ekstraksi Byte

#include <stdio.h>

int getByte(int x, int n)
{
    return (x >> (n * 8)) & 0xFF;
}

int main()
{
    int result = getByte(0x12345678, 1);
```

```
    printf("Result: 0x%.2X\n", result);
    return 0;
}
```

2. Hasil Eksekusi File

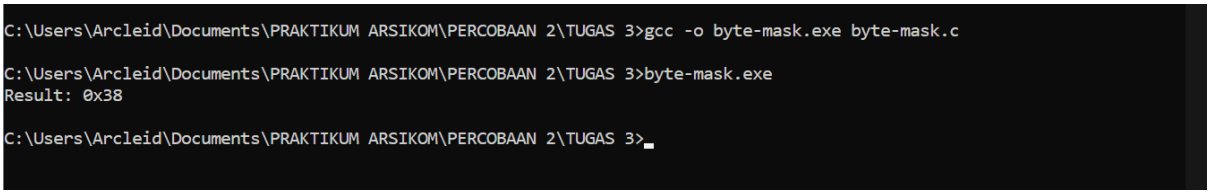


- **Tugas 3 : Fungsi Masking Byte**

1. Source Code

```
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : byte-mask.c
// Deskripsi       : Membuat Program Masking Byte

#include <stdio.h>

int bitMask(int high, int low)
{
    if (high < low)
      {
        return 0;
    }
    if (high > 31 || low > 31 || high < 0 || low < 0)
      {
        return 0;
    }
    return ((1 << (high - low + 1)) - 1) << low;
}

int main()
{
    int result = bitMask(5, 3);
    printf("Result: 0x%.2X\n", result);
    return 0;
}
```

2. Hasil Eksekusi



- **Tugas 4 : Fungsi Membalik Urutan Byte**

1. Source Code

```
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
```

```c
// Percobaan        : 2
// Tanggal          : 22 April 2024
// Nama (NIM)       : Akbar Dwi Herlambang (2308979)
// Nama File        : rev-byte.c
// Deskripsi        : Membuat Program Reverse Byte

#include <stdio.h>

int reverseBytes(int x)
{
    return ((x & 0xFF) << 24) | (((x >> 8) & 0xFF) << 16) | (((x >>
16) & 0xFF) << 8) | ((x >> 24) & 0xFF);
}

int main()
{
    int result = reverseBytes(0x01020304);
    printf("Result: 0x%.2X\n", result);
    return 0;
}
```

2. Hasil Eksekusi

```
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 4>gcc -o rev-byte.exe rev-byte.c

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 4>rev-byte.exe
Result: 0x4030201

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 4>
```

- **Tugas 5 : Fungsi Pengurangan Byte**

1. Source Code

```c
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul            : 1
// Percobaan        : 2
// Tanggal          : 22 April 2024
// Nama (NIM)       : Akbar Dwi Herlambang (2308979)
// Nama File        : min-byte.c
// Deskripsi        : Membuat Program Pengurangan Byte

#include <stdio.h>

int minBytes(int x, int y) {
    // Ambil byte paling rendah dari x
    int byteX = x & 0xFF;

    // Ambil byte paling rendah dari y
    int byteY = y & 0xFF;

    // Kembalikan nilai minimum dari kedua byte
    return byteX < byteY ? byteX : byteY;
}

int main()
{
    int x = 0x15;
    int y = 0x07;

    int result = minBytes(x, y);

    printf("minBytes(0x%.8X): 0x%.8X\n", x, result);
```

```
    return 0;
}
```

2. Hasil Eksekusi

```
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 5>gcc -o min-byte.exe min-byte.c

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 5>min-byte.exe
minBytes(0x00000015): 0x00000007
```

- **Tugas 6 : Fungsi Shift Register**

1. Source Code

```
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : shift-reg.c
// Deskripsi       : Membuat Program Fungsi Shift Register

#include <stdio.h>

unsigned int global_var = 0x00000000;

void inisialisasi()
{
    global_var = 0x00000000;
}

int shiftRegister(int x)
{
    global_var = (global_var << 5) | (x & 0x1F);
    return global_var;
}

int main()
{
    inisialisasi();

    printf("shiftRegister(0x04): 0x%.8X\n", shiftRegister(0x04));

    printf("shiftRegister(0x13): 0x%.8X\n", shiftRegister(0x13));

    return 0;
}
```

2. Hasil Eksekusi

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 6>gcc -o shift-reg.exe shift-reg.c

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 6>shift-reg.c

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 6>shift-reg.exe
shiftRegister(0x04): 0x00000004
shiftRegister(0x13): 0x00000093

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 6>
```

- **Tugas 7 : Program Enkripsi Sederhana**

1. Source Code Enkripsi

```
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : encrypt.c
// Deskripsi       : Membuat Program Enkripsi

#include <stdio.h>
#include "enc.h"

int encrypt(int input, int key)
{
    int byte1 = input & 0xFF;
    int byte2 = (input >> 8) & 0xFF;
    int byte3 = (input >> 16) & 0xFF;
    int byte4 = (input >> 24) & 0xFF;

    byte1 ^= key;
    byte2 ^= key;
    byte3 ^= key;
    byte4 ^= key;

    return (byte4 << 24) | (byte3 << 16) | (byte2 << 8) | byte1;
}
```

2. Source Code Dekripsi

```
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : byte-ext.c
// Deskripsi       : Membuat Program Dekripsi

#include <stdio.h>
#include "dec.h"

int decrypt(int encrypted, int key)
{
    int byte1 = encrypted & 0xFF;
    int byte2 = (encrypted >> 8) & 0xFF;
    int byte3 = (encrypted >> 16) & 0xFF;
    int byte4 = (encrypted >> 24) & 0xFF;

    byte1 ^= key;
    byte2 ^= key;
    byte3 ^= key;
    byte4 ^= key;

    return (byte4 << 24) | (byte3 << 16) | (byte2 << 8) | byte1;
}
```

3. Source Code Main Program

```
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
```

```c
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : main-program.c
// Deskripsi       : Kompilasi dari Program Enkripsi dan Dekripsi

#include <stdio.h>
#include "enc.h"
#include "dec.h"

int main()
{
    int input = 123456789;
      int key = 85;

    int encrypted = encrypt(input, key);
    printf("Enkripsi: %d\n", encrypted);

    int decrypted = decrypt(encrypted, key);
    printf("Dekripsi: %d\n", decrypted);

    return 0;
}
```

4. Makefile

```
all: simple-pass.exe

simple-pass.exe: encrypt.o decrypt.o main-program.o
	gcc encrypt.o decrypt.o main-program.o -o simple-pass.exe

encrypt.o: encrypt.c
	gcc -c encrypt.c

decrypt.o: decrypt.c
	gcc -c decrypt.c

main-program.o: main-program.c
	gcc -c main-program.c
```

5. Hasil Eksekusi

```
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 7>mingw32-make -f makefile
gcc -c encrypt.c
gcc -c decrypt.c
gcc -c main-program.c
gcc encrypt.o decrypt.o main-program.o -o simple-pass.exe

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 7>simple-pass.exe
Enkripsi: 1376688192
Dekripsi: 123456789

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 7>
```

- **Tugas 8 : Pointer dalam Assembly**

1. Gambaran coba.s pada Assembly

```
	.file "coba.c"
	.text
	.globl	_coba
	.def	_coba;	.scl	2;	.type 32;	.endef
_coba:
	pushl %ebp
	movl  %esp, %ebp
	subl  $16, %esp
```

```
        movl    12(%ebp), %eax
        movl    (%eax), %eax
        movl    %eax, -4(%ebp)
        movl    16(%ebp), %eax
        movl    (%eax), %eax
        movl    %eax, -8(%ebp)
        movl    -12(%ebp), %edx
        movl    -4(%ebp), %eax
        addl    %edx, %eax
        movl    %eax, -16(%ebp)
        movl    12(%ebp), %eax
        movl    -16(%ebp), %edx
        movl    %edx, (%eax)
        movl    16(%ebp), %eax
        movl    -4(%ebp), %edx
        movl    %edx, (%eax)
        movl    8(%ebp), %eax
        movl    -8(%ebp), %edx
        movl    %edx, (%eax)
        nop
        leave
        ret
        .ident      "GCC: (tdm-1) 5.1.0"
```

Dari yang saya pahami dari kode diatas, dapat disimpulkan bahwa nilai-nilai seperti '-8(%ebp), '-12(%ebp)' atau '-16(%ebp)' mereka sedang menunjuk kepada base pointer EBP dan sedang menggunakan memory yang berjarak sesuai dengan byte nilai tersebut.

3. Gambaran coba1.s pada Assembly

```
        .file "coba1.c"
        .text
        .globl      _coba
        .def    _coba;      .scl    2;      .type 32;    .endef
_coba:
        pushl %ebp
        movl    %esp, %ebp
        subl    $48, %esp
        movl    12(%ebp), %eax
        movl    (%eax), %eax
        movl    %eax, -44(%ebp)
        fildl -44(%ebp)
        fstpl -8(%ebp)
        movl    16(%ebp), %eax
        movl    (%eax), %eax
        movl    %eax, -44(%ebp)
        fildl -44(%ebp)
        fstpl -16(%ebp)
        fldl    -24(%ebp)
        faddl -8(%ebp)
        fstpl -32(%ebp)
        fldl    -32(%ebp)
        fnstcw      -34(%ebp)
        movzwl      -34(%ebp), %eax
        movb    $12, %ah
        movw    %ax, -36(%ebp)
        fldcw -36(%ebp)
        fistpl      -40(%ebp)
```

```
        fldcw -34(%ebp)
        movl  -40(%ebp), %edx
        movl  12(%ebp), %eax
        movl  %edx, (%eax)
        fldl  -8(%ebp)
        fldcw -36(%ebp)
        fistpl        -40(%ebp)
        fldcw -34(%ebp)
        movl  -40(%ebp), %edx
        movl  16(%ebp), %eax
        movl  %edx, (%eax)
        fldl  -16(%ebp)
        fldcw -36(%ebp)
        fistpl        -40(%ebp)
        fldcw -34(%ebp)
        movl  -40(%ebp), %edx
        movl  8(%ebp), %eax
        movl  %edx, (%eax)
        nop
        leave
        ret
        .ident        "GCC: (tdm-1) 5.1.0"
```

Perbedaan selain dari kode nya yang menjadi lebih banyak, yaitu adanya penggunaan instruksi-instruksi baru seperti fildl, fstpl, fldl, faddl, fnstcw, movzwl, movb, fldcw, dan fistpl, yang mana setelah di teliti lebih lanjut instruksi-instruksi tersebut digunakan untuk atau berkaitan dengan operasi floating point.

- **Tugas 9 : Fungsi Membalik Urutan Array**

1. Source Code

```
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : inverted-array.c
// Deskripsi       : Membuat Program Pembalik Array


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

int main() {
    char characters[100];
    char ch;
    int count = 0;

    printf("Masukkan beberapa karakter (tekan Enter dua kali untuk
selesai):\n");
    while (count < 100)
      {
        ch = getchar();
        if (ch == '\n')
            {
            ch = getchar();
            if (ch == '\n')
```

```
                {
                    break;
                } else
                    {
                        characters[count++] = '\n';
                    }
            }
            characters[count++] = ch;
        }

        printf("Karakter yang dimasukkan secara terbalik:\n");
        for (int i = count - 1; i >= 0; i--) {
            putchar(characters[i]);
        }
        printf("\n");

        return 0;
    }
```

2. Hasil Eksekusi inverted-array.exe

```
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 9>inverted-array.exe
Masukkan beberapa karakter (tekan Enter dua kali untuk selesai):
RAWR

Karakter yang dimasukkan secara terbalik:
RWAR

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 9>
```

- **Tugas 10 : Matriks Nama**

1. Source Code

```
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : matrix-name.c
// Deskripsi       : Membuat Program Matriks Nama

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

#define MAX_ROWS 10
#define MAX_COLS 50

int main() {
    char nama_orang[MAX_ROWS][MAX_COLS];
    int num_names;

    printf("Masukkan jumlah nama orang: ");
    scanf("%d", &num_names);
    getchar();

        printf("Masukkan nama orang satu per satu:\n");
    for (int i = 0; i < num_names; i++) {
        printf("Nama orang %d: ", i + 1);
        fgets(nama_orang[i], MAX_COLS, stdin);
    }

    printf("\nMatriks nama_orang yang dimasukkan:\n");
```

```
    for (int i = 0; i < num_names; i++) {
        printf("%d: %s", i + 1, nama_orang[i]);
    }

    return 0;
}
```

2. Hasil Eksekusi matrix-name.exe

```
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 10>matrix-name.exe
Masukkan jumlah nama orang: 5
Masukkan nama orang satu per satu:
Nama orang 1: Aron
Nama orang 2: Ijay
Nama orang 3: Nurdin
Nama orang 4: Amar
Nama orang 5: Rehan

Matriks nama_orang yang dimasukkan:
1: Aron
2: Ijay
3: Nurdin
4: Amar
5: Rehan

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 10>
```

- **Tugas 11 : Matriks dengan Pointer**

```
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : matrix-point.c
// Deskripsi       : Membuat Program Matriks Pointer

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

int main() {
    int jumlah_nama;
    printf("Masukkan jumlah nama orang: ");
    scanf("%d", &jumlah_nama);

    char *nama_orang[jumlah_nama];

    printf("Masukkan nama orang satu per satu:\n");
    for (int i = 0; i < jumlah_nama; i++)
      {
        nama_orang[i] = (char *)malloc(50 * sizeof(char));

        printf("Nama orang %d: ", i + 1);
        scanf("%s", nama_orang[i]);
    }

    printf("\nMatriks nama_orang yang dimasukkan:\n");
    for (int i = 0; i < jumlah_nama; i++)
      {
        printf("%d: %s\n", i + 1, nama_orang[i]);
        free(nama_orang[i]);
    }

    return 0;
}
```

2. Hasil Eksekusi matrix-point.exe

```
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 11>gcc -o matrix-point.exe matrix-point.c

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 11>matrix-point.exe
Masukkan jumlah nama orang: 3
Masukkan nama orang satu per satu:
Nama orang 1: kow
Nama orang 2: lou
Nama orang 3: arr

Matriks nama_orang yang dimasukkan:
1: kow
2: lou
3: arr

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 11>
```

Dari yang saya pahami, penggunaan memori tidak jauh berbeda, untuk mendapatkan alokasi memori yang fleksibel kita perlu juga untuk menyesuaikan Panjang nama yang dimasukkan.

- **Tugas 12 : Perkalian Matriks**

```c
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : kali-matrixs.c
// Deskripsi       : Membuat Program Perkalian Matrix

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

struct Matriks
{
    int jumlahBaris;
    int jumlahKolom;
    int **nilai;
};

int** alloc(int baris, int kolom)
{
    int **matriks = (int **)malloc(baris * sizeof(int *));
    for (int i = 0; i < baris; i++)
      {
        matriks[i] = (int *)malloc(kolom * sizeof(int));
    }
    return matriks;
}

void input(struct Matriks *matriks)
{
    printf("Masukkan matriks %d x %d:\n", matriks->jumlahBaris,
matriks->jumlahKolom);
    for (int i = 0; i < matriks->jumlahBaris; i++)
      {
        for (int j = 0; j < matriks->jumlahKolom; j++)
            {
            scanf("%d", &matriks->nilai[i][j]);
        }
```

```
    }
}

struct Matriks* mulMatriks(struct Matriks *A, struct Matriks *B)
{
    if (A->jumlahKolom != B->jumlahBaris)
      {
        printf("Perkalian matriks tidak dapat dilakukan. Jumlah
kolom matriks A harus sama dengan jumlah baris matriks B.\n");
        return NULL;
    }

    struct Matriks *hasil = (struct Matriks *)malloc(sizeof(struct
Matriks));
    hasil->jumlahBaris = A->jumlahBaris;
    hasil->jumlahKolom = B->jumlahKolom;
    hasil->nilai = alloc(hasil->jumlahBaris, hasil->jumlahKolom);

    for (int i = 0; i < A->jumlahBaris; i++)
      {
        for (int j = 0; j < B->jumlahKolom; j++)
            {
            hasil->nilai[i][j] = 0;
            for (int k = 0; k < A->jumlahKolom; k++)
                  {
                hasil->nilai[i][j] += A->nilai[i][k] *
B->nilai[k][j];
            }
        }
    }

    return hasil;
}

void show_matr(struct Matriks *matriks)
{
    printf("Matriks %d x %d:\n", matriks->jumlahBaris,
matriks->jumlahKolom);
    for (int i = 0; i < matriks->jumlahBaris; i++)
      {
        for (int j = 0; j < matriks->jumlahKolom; j++)
            {
            printf("%d ", matriks->nilai[i][j]);
        }
        printf("\n");
    }
}

void del_matr(struct Matriks *matriks)
{
    for (int i = 0; i < matriks->jumlahBaris; i++)
      {
        free(matriks->nilai[i]);
    }
    free(matriks->nilai);
    free(matriks);
}

int main()
{
    struct Matriks A, B;
```

```
    printf("Masukkan ukuran matriks A (baris kolom): ");
    scanf("%d %d", &A.jumlahBaris, &A.jumlahKolom);
    printf("Masukkan ukuran matriks B (baris kolom): ");
    scanf("%d %d", &B.jumlahBaris, &B.jumlahKolom);

    A.nilai = alloc(A.jumlahBaris, A.jumlahKolom);
    B.nilai = alloc(B.jumlahBaris, B.jumlahKolom);

    input(&A);
    input(&B);

    struct Matriks *hasil = mulMatriks(&A, &B);
    if (hasil != NULL)
      {
        show_matr(hasil);
        del_matr(hasil);
    }

    for (int i = 0; i < A.jumlahBaris; i++)
      {
        free(A.nilai[i]);
    }
    free(A.nilai);
    for (int i = 0; i < B.jumlahBaris; i++)
      {
        free(B.nilai[i]);
    }
    free(B.nilai);

    return 0;
}
```

2. Hasil Eksekusi kali-matriks.exe

```
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 12>kali-matriks.exe
Masukkan ukuran matriks A (baris kolom): 3 3
Masukkan ukuran matriks B (baris kolom): 3 3
Masukkan matriks 3 x 3:
1 2 3
4 5 6
7 8 9
Masukkan matriks 3 x 3:
1 2 3
4 5 6
7 8 9
Matriks 3 x 3:
30 36 42
66 81 96
102 126 150

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 12>
```

- **Tugas 13 : Penjumlah Biner dengan Array**

1. Source Code show-bit.c. dan Header Filenya

```
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul          : 1
// Percobaan      : 2
// Tanggal        : 22 April 2024
// Nama (NIM)     : Akbar Dwi Herlambang (2308979)
// Nama File      : show-bit.c
// Deskripsi      : Membuat Program Penampil Array Bit

#include <stdio.h>
#include "show-bit.h"
```

```c
void showArray(int array[], int panjang)
{
    for (int i = panjang - 1; i >= 0; i--) {
        printf("%d", array[i]);
    }
    printf("\n");
}
```

```c
#ifndef SHOW_ARRAY
#define SHOW_ARRAY 100

      void showArray(int array[], int panjang);

#endif
```

2. Source Code plus-bit.c dan Header Filenya

```c
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : plus-bit.c
// Deskripsi       : Membuat Program Penjumlahan Bit

#include <stdio.h>
#include "plus-bit.h"

void plusBit(int a[], int b[], int hasil[], int panjang)
{
    int carry = 0;
    for (int i = 0; i < panjang; i++) {
        int jumlah = a[i] + b[i] + carry;
        hasil[i] = jumlah % 2;
        carry = jumlah / 2;
    }
}
```

```c
#ifndef PLUS_BIT
#define PLUS_BIT 100

      void plusBit(int a[], int b[], int hasil[], int panjang);

#endif
```

3. Source Code minus-bit.c dan Header Filenya

```c
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : minus-bit.c
// Deskripsi       : Membuat Program Pengurangan Bit

#include <stdio.h>
#include "minus-bit.h"
```

```c
void minusBit(int a[], int b[], int hasil[], int panjang)
{
    int borrow = 0;
    for (int i = 0; i < panjang; i++) {
        int selisih = a[i] - b[i] - borrow;
        if (selisih < 0) {
            hasil[i] = selisih + 2;
            borrow = 1;
        } else {
            hasil[i] = selisih;
            borrow = 0;
        }
    }
}
```

```c
#ifndef MINUS_BIT
#define MINUS_BIT 100

      void minusBit(int a[], int b[], int hasil[], int panjang);

#endif
```

4. Source Code main-bit.c

```c
// Organisisi dan Arsitektur Komputer Tekom Kelas 2C
// Modul           : 1
// Percobaan       : 2
// Tanggal         : 22 April 2024
// Nama (NIM)      : Akbar Dwi Herlambang (2308979)
// Nama File       : kali-matrixs.c
// Deskripsi       : Kompilasi dari Penampil, Penjumlahan, dan
Pengurangan Bit

#include <stdio.h>
#include "show-bit.h"
#include "plus-bit.h"
#include "minus-bit.h"

int main()
{
    int panjang = 8;

    int a[] = {1, 1, 1, 0, 0, 0, 1, 1};
    int b[] = {1, 1, 1, 1, 0, 0, 1, 0};

    printf("Angka 103 dalam representasi bit: ");
    showArray(a, panjang);

    printf("Angka 114 dalam representasi bit: ");
    showArray(b, panjang);

    int hasilPenjumlahan[panjang];
    plusBit(a, b, hasilPenjumlahan, panjang);
    printf("Hasil penjumlahan: ");
    showArray(hasilPenjumlahan, panjang);

    int hasilPengurangan[panjang];
    minusBit(a, b, hasilPengurangan, panjang);
    printf("Hasil pengurangan: ");
```

```
    showArray(hasilPengurangan, panjang);

    return 0;
}
```

5. Makefile

```
all: operasi-bit.exe

operasi-bit.exe: show-bit.o plus-bit.o minus-bit.o main-bit.o
	gcc show-bit.o plus-bit.o minus-bit.o main-bit.o -o operasi-
bit.exe

show-bit.o: show-bit.c
	gcc -c show-bit.c

plus-bit.o: plus-bit.c
	gcc -c plus-bit.c

minus-bit.o: minus-bit.c
	gcc -c minus-bit.c

main-bit.o: main-bit.c
	gcc -c main-bit.c
```

6. Hasil Eksekusi

```
C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 13>mingw32-make -f makefile
gcc -c main-bit.c
gcc show-bit.o plus-bit.o minus-bit.o main-bit.o -o operasi-bit.exe

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 13>operasi-bit.exe
Angka 103 dalam representasi bit: 11000111
Angka 114 dalam representasi bit: 01001111
Hasil penjumlahan: 00010110
Hasil pengurangan: 01111000

C:\Users\Arcleid\Documents\PRAKTIKUM ARSIKOM\PERCOBAAN 2\TUGAS 13>
```

- **Hasil dan Analisis**

  Dari jawaban-jawaban tersebut, kita dapat menarik beberapa kesimpulan penting tentang konsep-konsep dasar dalam pemrograman dan representasi data:

  Tipe Data dan Representasinya: Berbagai tipe data, seperti float dan double, memiliki representasi yang khas dalam memory, seperti IEEE-754 untuk tipe data float dan double.

  Struktur Data: Urutan penulisan dalam sebuah struktur data mempengaruhi ukuran struktur tersebut karena padding yang ditambahkan oleh compiler.

  Pointer: Pointer digunakan untuk menyimpan alamat memori dan memberikan fleksibilitas dalam mengakses dan memanipulasi data. Namun, penggunaan yang tidak benar dapat menyebabkan berbagai masalah, seperti segmentation faults dan memory leaks.

  Array: Array adalah kumpulan elemen data yang sejenis. Ada perbedaan antara array dinam Pemrograman komputer membutuhkan pemahaman yang mendalam tentang representasi data, struktur data, dan berbagai operasi yang dapat dilakukan terhadapnya. Dari berbagai pertanyaan di atas, kita dapat melihat bahwa pemahaman tentang tipe

data dan representasinya dalam memory, seperti float dan double, sangat penting. Selain itu, konsep struktur data juga sangat relevan, terutama dalam hal penggunaan pointer. Pointer memungkinkan pengaksesan fleksibel terhadap data dengan menyimpan alamat memori, namun penggunaannya harus hati-hati untuk menghindari kesalahan yang berpotensi fatal, seperti segmentation faults.

Array juga merupakan struktur data fundamental dalam pemrograman, dan pemahaman tentang perbedaan antara array dinamis dan array statis sangatlah penting dalam alokasi memory. Operasi bitwise juga memainkan peran penting dalam manipulasi data pada level bit, yang dapat digunakan untuk berbagai keperluan, mulai dari kriptografi hingga optimasi performa. Perkalian matriks adalah contoh konkret dari penggunaan algoritma yang tepat untuk meningkatkan efisiensi dan kinerja.

Selain itu, pemahaman tentang sistem bilangan two's complement dan operasi pengurangan dalam konteks ini penting untuk penanganan data dengan sign bit. Pemahaman tentang penggunaan pointer dalam bahasa assembly juga penting, karena pointer ke tipe data yang berbeda akan menunjuk pada lokasi memori yang berbeda sesuai dengan ukuran dari tipe data yang ditunjuk.