Percobaan II
Pointer, Structure, Array, dan Operasi dalam Level Bit

Ahmad Aziz (13220034)
Asisten: Aditya Anandita Dharma Putra (13219043)
Tanggal Percobaan : 23/09/2022
EL3111 Praktikum Arsitektur Sistem Komputer
Laboratorium Sinyal dan Sistem – Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Abstrak*— Pada praktikum modul kedua ini yaitu modul Pointer, Structure, Array, dan Operasi dalam Level Bit akan dilakukan beberapa percobaan dengan topik mengenai tipe data, operator bitwise dalam bahasa C, structure, array, dan juga pointer. Pada praktikum ini ada 13 percobaan yang akan dilakukan. Semua percobaan yang pada praktikum ini akan dilakukan pada computer dengan sistem operasi windows 11, dengan compiler program bahasa C menggunakan GCC serta teks editor yang digunakan adalah Visual Studio Code dan notepad++. Percobaan pertama yang dilakukan adalah fungsi XOR, pada percobaan ini akan digunakan operasi pade level bitwise dalam bahasa C yaitu operator XOR dimana untuk membuat fungsi ini hanya dengan operator AND dan operator NOT. Percobaan kedua adalah membuat fungsi enkstraksi byte, pada percobaan ini akan membuat sebuah fungsi yang mengebalikan nilai byte ke-n pada suatu value. Percobaan senlanjutnya masih berkaitan dengan byte yaitu membuat fungsi masking byte. Fungsi ini menghasilkan mask pada bit diantara rentang atas dan bawah yang dimasukkan pada parameter fungsi. Percobaan kelima adalah fungsi pengurangan dimana percobaan ini masih berkaitan dengan bitwise operator dimana untuk melakukan pengurangan hanya dilakukan dengan operator penjumlahan dan invers saja. Percobaan ke-6 sampai percobaan ke-7 masih berkaitan dengan bitwise operator yaitu membuat fungsi shift register dan enkripsi dalam level bitwise. Dari percobaan ke-8 hingga percobaan terakhir ke-13 dilakukan percobaan mengenai pointer dan juga mengenai array. Tujuan praktikum ini adalah membuat semua fungsi dan kode yang berfungsi sesuai dengan spesifikasi agar praktikan dapat memahami materi pada percobaan dengan baik.

***Kata Kunci—Struct, pointer, array,* bitwise.**

I. Pendahuluan

Pada praktikum modul 2 yaitu Pointer, Structure, Array, Dan Operasi Dalam Level Bit dilakukan sebanyak 13 percobaan yang bertujuan diantaranya sebagai berikut:

- Praktikan memahami representasi informasi dalam level bit yang disimpan pada memory.
- Praktikan mampu menggunakan operator-operator bitwise dalam bahasa C untuk mengolah informasi yang tersimpan dalam memory.
- Praktikan memahami fungsi pointer dan mampu menggunakan pointer untuk melakukan pengolahan data di dalam memory.
- Praktikan memahami array beserta representasinya dalam memory dan pengolahan informasinya dalam bahasa C.
- Praktikan memahami structure beserta representasinya dalam memory dan pengolahannya dalam bahasa C.

Praktikum pada modul ini menggunakan bahasa pemrograman C dengan compiler GCC untuk melakukan percobaan dan juga bahasa assembly sebagai hasil compiling dari bahasa C yang akan dianalisis. Ada beberapa topik yang dibahas pada modul praktikum ini yaitu sebagai berikut:

a. Tipe data.
b. Operator Bitwise dalam Bahasa C.
c. Structure.
d. Array.
e. Pointer.

Dalam melakukan percobaan dan analisis pada praktikum modul ini, perangkat lunak dan alat yang digunakan adalah sebagai berikut:

a. Compiler GCC
b. CodeBlocks
c. Code editor Visual Studio Code
d. HexEdit

II. Landasan Teoretis

Bahasa pemrograman C merupakan bahasa pemrograman yang bersifat processor independent. Artinya bahasa pemrograman C dapat berjalan tanpa bergantung pada jenis ataupun arsitektur processor dimana program tersebut dijalankan. Karena itu, bahasa C dapat dijalankan pada berbagai mesin tanpa mengubah kodenya. Karena flexibilitas dan portabilitasnya inilah bahasa pemrograman C banyak dipakai dalam pemrograman sistem ebeded. Bahasa pemrograman C dapat berjalan pada berbagai mesin karena compiler yang mendukung pada setiap mesin yang menjalankannya. Mesin tidak dapat membaca bahasa C, mesin hanya dapat membaca bahasa mesin (binary). Oleh karena itu, dibutuhkan sebuah

proses agar bahasa C dapat dibaca oleh mesin, proses ini disebut proses kompilasi.

A. *Tipe Data*

Tipe data merupakan representasi data yang disimpan dalam memory. Tipe data menentukan operasi yang dapat dilakukan pada suatu data bertipe tertentu, rentang nilai yang mungkin dimiliki oleh data bertipe tertentu, arti dari data, dan cara menyimpan data tersebut dalam memory. Terdapat beberapa tipe data yang telah tersedia dalam bahasa C. Tipe data yang tersedia ini disebut simple data type.

Masing-masing tipe data memiliki ukuran yang berbeda-beda untuk disimpan di dalam memory. Dalam bahasa C, kita dapat menggunakan sintaks sizeof untuk mengetahui besar tipe data tersebut di dalam memory. Contohnya, untuk mengetahui ukuran tipe data integer, kita cukup menggunakan perintah sizeof(int) untuk mengetahui ukurannya. Kita juga dapat melihat rentang nilai yang direpresentasikan oleh masing-masing tipe data.

Tipe Data	Ukuran	Rentang Nilai
Char	1 Byte	-128 – 127
Unsigned Char	1 Byte	0 – 255
Signed Char	1 Byte	-128 – 127
Short	2 Byte	-32768 – 32767
Unsigned Short	2 Byte	0 – 65535
Signed Short	2 Byte	-32768 – 32767
Int	4 Byte	-2147483648 – 2147483647
Unsigned Int	4 Byte	0 – 4294967295
Signed Int	4 Byte	-2147483648 – 2147483647
Long	4 Byte	-2147483648 – 2147483647
Unsigned Long	4 Byte	0 – 4294967295
Signed Long	4 Byte	-2147483648 – 2147483647
Long Long	8 Byte	-9223372036854775808 – 9223372036854775807
Unsigned Long Long	8 Byte	0 – 18446744073709551615
Signed Long Long	8 Byte	9223372036854775808 – 9223372036854775807
Float	4 Byte	?
Double	8 Byte	?

Data pada tabel berikut dapat diketahui dengan program sederhana sebagai berikut. Perhatikan bahwa mesin berbeda dapat memberikan hasil yang berbeda berkaitan dengan besar tipe data. Dengan adanya perbedaan ukuran masing-masing tipe data, diperlukan sebuah mekanisme alignment pada memory agar setiap data tersusun dengan baik di dalam memory dan dapat diproses oleh mikroprosesor. Dengan alignment, data-data variabel disimpan dalam lokasi memory yang memiliki address offset yang berupa kelipatan dari ukuran word. Hal ini akan menambah performance karena data disusun sesuai cara mikroprosesor menggunakan memory.

B. *Operator Bitwise dalam Bahasa C*

Bahasa C mendukung pengolahan informasi dalam level bit menggunakan operator bitwise. Berbeda dengan operator level byte, operator bitwise akan mengoperasikan data untuk setiap bit. Sedangkan operator level byte, data akan diolah dalam bentuk 1 byte (1 byte = 8 bit). Operator bitwise dapat digunakan pada berbagai tipe data seperti char, int, short, long, atau unsigned. Operator-operator bitwise dalam bahasa C didefinisikan sebagai berikut.

Simbol	Operator
&	Bitwise AND
|	Bitwise inclusive OR
^	Bitwise exclusive OR
<<	Left Shift
>>	Right Shift
~	Bitwise NOT (one's complement) (unary)

Bahasa C juga memiliki operator logika AND, inclusive OR, dan NOT. Operator ini sering tertukar dengan operator bitwise. Operator logika tersebut diberikan sebagai berikut. Pada operasi logika, setiap argumen bukan nol merepresentasikan TRUE, sedangkan argumen nol merepresentasikan FALSE. Ekspresi logika akan mengembalikan nilai 1 untuk TRUE dan nilai 0 untuk FALSE.

Simbol	Operator
&&	Logical AND
||	Logical OR
!	Logical NOT

Khusus untuk operator Right Shift, terdapat dua jenis operator Right Shift, yaitu Logical Right Shift dan Arithmetic Right Shift. Logical Right Shift akan mengisi bit MSB dengan nilai 0 sementara Arithmetic Right Shift akan mengisi bit MSB sesuai dengan tanda (sign) variabel tersebut menurut aturan two's complement. Untuk Left Shift, tidak ada perbedaan antara Logical Left Shift dan Arithmetic Left Shift. Pada umumnya, seluruh mesin dapat mendukung dua jenis operator Right Shift dan Left Shift ini.

C. *Structure*

Structure (struct) merupakan complex data type yang mendefinisikan daftar variabel yang akan ditempatkan dalam blok memory menggunakan satu nama. Dengan demikian, setiap variabel berbeda pada structure dapat diakses menggunakan sebuah single pointer atau dengan menggunakan nama structure itu sendiri. Pada structure, masing-masing variabel disimpan dalam blok memory yang kontigu yang biasanya memiliki delimiter berupa panjang word. Kita juga dapat menggunakan sintaks sizeof untuk memeriksa ukuran structure yang kita definisikan. Perlu diingat bahwa bahasa C tidak mengizinkan kita melakukan deklarasi rekursif terhadap structure (sebuah structure tidak boleh berisi structure bertipe yang sama dengan structure tersebut). Kita dapat menggunakan pointer untuk melakukannya. Beberapa mesin 33 juga membutuhkan alignment data pada memory secara spesifik sehingga ukuran structure dapat berbeda karena compiler secara otomatis melakukan alignment data-data pada structure, contohnya dengan padding.

D. *Array*

Array merupakan kumpulan lokasi penyimpanan data yang kontigu (berurutan) dengan tipe data yang sama. Setiap lokasi penyimpanan dalam sebuah array disebut elemen array. Array dialokasikan secara sekaligus dalam memory sesuai dengan ukurannya. Karena letak elemen yang berurutan, akses elemen array pada memory relatif lebih mudah dan cepat dibandingkan dengan struktur data LinkedList. Setiap elemen dalam array dapat diakses menggunakan indeks yang biasanya berupa bilangan bulat skalar bukan negatif. Dalam bahasa C, elemen

pertama dalam array diberi indeks 0. Representasi array dalam memory dengan deklarasi int nim[8] adalah sebagai berikut (asumsikan address elemen ke-0 adalah 0x4000 dengan nilai elemen pertama adalah 1, nilai elemen kedua adalah 3, nilai elemen ketiga adalah 2, nilai elemen keempat adalah 1, nilai elemen kelima adalah 1, nilai elemen keenam adalah 0, nilai elemen ketujuh adalah 0, dan nilai elemen kedelapan adalah 7).

Nilai	1	3	2	1	1	0	0	7
Alamat	0x4000	0x4004	0x4008	0x400C	0x4010	0x4014	0x4018	0x401C
Indeks	nim[0]	nim[1]	nim[2]	nim[3]	nim[4]	nim[5]	nim[6]	nim[7]

Bahasa C mendukung deklarasi array secara statis maupun secara dinamis. Array statis memiliki ukuran yang tidak bisa diubah-ubah sedangkan array dinamis memiliki ukuran yang dapat ditentukan saat program sedang berjalan. Array dinamis dapat dideklarasikan dengan contoh int pusheen[]. Dengan demikian pusheen merupakan array yang memiliki elemen bertipe integer namun dengan banyak elemen yang belum didefinisikan. Untuk melakukan alokasi terhadap array pusheen, kita dapat menggunakan perintah malloc atau calloc dalam bahasa C. Untuk mengubah ukuran array dinamis yang telah dialokasikan, kita dapat menggunakan perintah realloc. Untuk melakukan dealokasi array dinamis, kita dapat menggunakan perintah free. Perhatikan bahwa pada beberapa kasus, perintah realloc dapat menyebabkan program tidak efisien contohnya saat array diubah ukurannya menjadi lebih besar dan sistem harus melakukan pemindahan elemen-elemen array ke posisi memory yang baru agar perbesaran ukuran array dapat dilakukan.

Array juga dapat memiliki elemen array berupa array. Dengan demikian, kita dapat mendefinisikan array n-dimensi. Contohnya, sebuah matriks merupakan array dengan dua dimensi. Array tersebut memiliki elemen array berupa array, contohnya int pusheen[][5] atau int pusheen[4][5]. Namun, karena memory komputer bersifat linear, komputer akan menyimpan array n-dimensi dalam bentuk linear juga. Hal ini menyebabkan kita harus memperhatikan urutan indeks untuk mengakses setiap elemen array n-dimensi karena hal ini akan berpengaruh terhadap performance dari program yang kita buat terlebih data array yang diolah cukup besar, contohnya seberapa baikkah program yang kita buat dalam memanfaatkan cache memory (cache-friendly).

E. Pointer

Pointer merupakan variabel yang menyimpan alamat memory. Dengan kata lain, pointer memiliki nilai alamat memory untuk melakukan referensi terhadap suatu objek yang tersimpan dalam memory komputer. Dengan menggunakan pointer, kita dapat memiliki akses terhadap memory secara langsung. Untuk setiap tipe data T, terdapat pointer ke T. Deklarasi pointer dalam bahasa C dapat dilakukan dengan mudah, contohnya int *ptr yang merupakan pointer yang melakukan referensi terhadap objek bertipe integer.

Pointer juga dapat digunakan untuk menunjukkan structure berdasarkan alamatnya di memory. Hal ini sangat berguna untuk melakukan passing structure ke atau dari sebuah fungsi hanya dengan memberikan alamat structure tersebut di memory. Pointer ini juga dapat di-dereferensi seperti halnya pointer lain dalam bahasa C, yaitu menggunakan operator dereference (*). Selain itu, juga terdapat operator yang sangat berguna yaitu struct_name -> member untuk melakukan dereferensi pointer-to-struct lalu mengakses nilai dari anggota structure tersebut. Operator tersebut memiliki ekuivalensi dengan (*struct_name).member. Sebetulnya, sebuah fungsi dapat langsung mengembalikan sebuah structure walaupun hal ini terkadang tidak efisien saat dijalankan.

Dalam array, indeks biasanya didefinisikan sebagai perhitungan matematika terhadap alamat pointer. Dengan demikian penulisan array[i] memiliki ekuivalensi dengan *(array + i). Perhitungan matematika terhadap pointer untuk mengakses setiap elemen array dapat dilakukan karena array memiliki elemen yang tersusun secara kontigu (berurutan tanpa jeda).

III. HASIL DAN ANALISIS

Setelah melakukan percobaan pada semua tugas didapatkan hasil sebagai brerikut:

A. Tugas I : Fungsi XOR

Pada percobaan tugas 1 ini praktikan membuat dan mengimplementasikan sebuah fungsi yang sudah diberikan yaitu fungsi XOR dalam bahasa C. Percobaan ini berkaitan dengan operasi bitwise. Untuk mengimplementasikan fungsi ini hanya diperbolehkan menggunakan operator bitwise yang sudah ditentukan yaitu operator AND dan operator NOT.

Berikut ini adalah fungsi yang diberikan dan diimplementasikan dalam bahasa C pada file header:

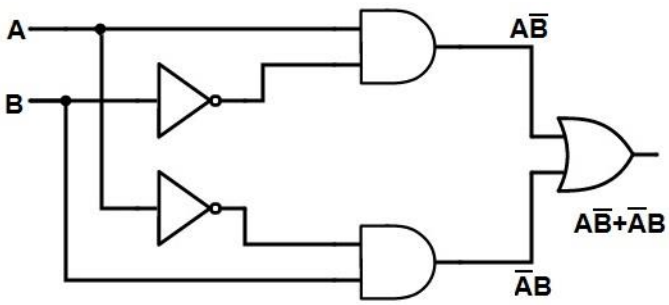
```
#include <stdio.h>
int bitXor (int x, int y);
```

Selanjutnya diimplementasikan dalam file C nya sebagai berikut:

```
#include "bitXor.h"

int bitXor(int x, int y) {
  return ~(~x & ~y) & ~(x & y);
}
```

Fungsi XOR diimplementasikan dengan menggunakan operator NOT dan AND. Jika kita gambarkan dalam gerbang logika dan aljabar boolean, fungsi XOR diatas dapat digambarkan dalam bentuk berikut:



Pada file main.c diinclude file header bitXor dan juga file header untuk fungsi printbitbyte:

```
#include "bitXor.h"
#include "printbitbyte/printbitbyte.h"
```

```
int main() {
    int x;
    int y;
    printf("Masukkan nilai x: ");
    scanf("%d", &x);
    printf("Masukkan nilai y: ");
    scanf("%d", &y);

    printf("x = %d", x);
printBit(sizeof(x), &x);
    printf("y = %d", y);
printBit(sizeof(y), &y);

    int z = bitXor(x, y);
    printf("z = %d", z);
printBit(sizeof(z), &z);
    return 0;
}
```

Pada program main tersebut user akan memasukkan nilai operan yang akan dilakukan operasi XOR. Hasil yang didapatkan pada program ini sesuai dengan nilai seharusnya dan juga test case pada modul.

B. Tugas 2 : Fungsi Ekstraksi Byte

Pada percobaan tugas 2, yaitu membuat fungsi untuk melakukan ekstraksi byte ke-n dari suatu data. Deklarasi fungsi diberikan dimasukkan pada file header beserta standard library untuk melakukan print dan input sebagai berikut:

```
#include <stdio.h>
int getByte (int x, int n);
```

Selanjutnya implementasi fungsi yang sudah dideklarasikan didalam file c untuk fungsi tersebut dengan include file header yang telah dibuat.

Berikut adalah kode implementasi ekstrasi byte:

```
#include "getByte.h"

int getByte(int x, int n){
    return (x >> (n << 3)) & 0xFF;
}
```

Pada implementasi fungsi getByte tersebut, sesuai dengan deklarasi fungsi yang diberikan di modul, fungsi menerima parameter berupa 2 integer. Integer x sebagai data yang akan diekstraksi dan n merupakan nomor byte yang akan diekstraksi. Untuk mendapatkan nilai ekstraksi, x geser ke kanan right shift sebanyak n yang diambil 8 bitnya atau 1-byte sehingga bisa mendapatkan byte yang diinginkan. Selanjutnya, kita masking dengan value 0xFF atau semua satu dalam 8-bit sehingga kita mendapatkan nilai 8 bit saya dan yang lain menjadi 0.

Fungsi tersebut digunakan untuk diuji pada file main.c. Untuk melakukan pengecekan dan analisis yang lebih detail, digunakan fungsi printbitbyte yang sudah dibuat pada tugas pendahuluan untuk melakukan print nilai dalam bit dan juga byte. Berikut ini adalah penggunaan fungsi pada main.c:

```
#include "getByte.h"
#include "printbitbyte/printbitbyte.h"

int main() {
    int x;
    int n;
    printf("Masukkan nilai x: ");
    scanf("%x", &x);
    printf("Masukkan nilai n: ");
    scanf("%d", &n);

    printf("Byte x = 0x ");
printByte((byte_pointer) &x, sizeof(x));

    int z = getByte(x, n);
    printf("byte ke %d = 0x%x", n, z);
    return 0;
}
```

Hasil eksekusi berjalan tanpa error dan mendapatkan hasil yang sesuai untuk melakukan ekstraksi byte dan sesuai juga dengan test case pada modul. Sehingga fungsi implementasi dengan operator bitwise untuk melakukan ekstraksi byte yang dibuat sudah benar.

C. Tugas 3 : Fungsi Masking Byte

Percobaan ketiga yaitu fungsi masking byte dimana fungsi ini menghasilkan masking untuk seluruh bit diantara batas bit pada data yang dimasukkan di parameter fungsi. Batas bawah dan batas atas diluarnya dibuat menjadi 0. Berikut adalah deklarasi fungsi yang akan diimplementasikan yang terdapat pada modul:

```
#include <stdio.h>

int bitMask (int highbit, int lowbit);
```

Deklarasi fungsi dilakukan di file header bitmask,h dimana fungsi ini mengembalikan nilai berupa integer yang nantinya sudah dilakukan masking. Parameter berupa 2 integer highbit dan lowbit. Digunakan juga standard library untuk melakukan input dan juga print pada terminal.

Fungsi kemudian diimplementasikan pada file bitmask.c dengan cara memanggil file header yang telah dibuat. Berikut ini adalah implementasi fungsi bitmask pada file c:

```
#include "bitMask.h"

int bitMask(int highbit, int lowbit){
    int mask = 0;
    int i;
    for (i = lowbit; i <= highbit; i++){
        mask |= (1 << i);
    }
    return mask;
}
```

Pertama, fungsi melakukan deklarasi integer bernilai 0 yang nantinya akan menjadi hasil masking. Kemudian dideklarasikan juga integer untuk increment looping dengan for untuk masking setiap bit.

Looping dilakukan dimulai dari lowbit hingga highbit. Selanjutnya masking dilakukan dengan operator OR terhadap 1 yang bernilai bit 1 pada LSBnya pada setiap bit yang berada dalam rentang loopnya dalam hal ini lowbit dan highbitnya. 1 akan terus digeser sesuai dengan increment pada loopingnya sehingga semua bit akan menjadi 1.

Fungsi tersebut selanjutnya diuji pada main file dengan memanggil fungsi bitmask tersebut. Pada main file dipanggil file header fungsi bitmask yang sudah dibuat. Digunakan juga fungsi printbitbyte pada percobaan sebelumnya untuk memperjelas hasil yang didapatkan dalam bit. Berikut ini adalah kode untuk main.c file:

```
#include "bitMask.h"
#include "printbitbyte/printbitbyte.h"

int main(){
    int highbit, lowbit;
    printf("Masukkan highbit: ");
    scanf("%d", &highbit);
    printf("Masukkan lowbit: ");
    scanf("%d", &lowbit);
    int mask = bitMask(highbit, lowbit);
    printf("Hasil mask = 0x%x = ", mask);
    printBit(sizeof(int), &mask);
}
```

Pada penggunaan fungsi pada main file ini, user memasukkan nilai untuk lowbit dan highbit yang disimpan dalam variable integer. Kemudian nilai tersebut dimasukkan dalam parameter fungsi pada pemanggilang fungsi dan hasil masking di tampilkan pada terminal dalam bentuk hex dan juga binary dengan menggunakan fungsi printBit.

Hasil output yang didapatkan sesuai dengan yang seharusnya dan dengan testcase pada modul sehingga fungsi yang sudah dibuat benar dan berjalan dengan baik.

D. Tugas 4: Fungsi Membalik Urutan Byte

Pada percobaan tugas 4 yaitu fungsi untuk membalik urutan byte. Fungsi menerima suatu data kemudian membalik setiap byte dengan mengubah urutan dari byte LSB ke MSB setiap bytenya (8 bit). Berikut adalah deklarasi fungsi yang diberikan pada modul yang diimplementasikan dalam file header reverseByte.h :

```
#include <stdio.h>
int reverseBytes (int x);
```

Fungsi mengembalikan nilai (return) berupa integer hasil reverse byte dengan perameter berupa interger nilai yang akan dibalik bytenya. Fungsi ini kemudian dibuat dalam file C dengan memanggil file header yang telah dibuat. Berikut ini adalah implementasi fungsi pada file reverseByte.c:

```
#include "reverseByte.h"

int reverseBytes(int x){
    int y = 0;
    int i;
    for (i = 0; i < 4; i++){
        y = y << 8;
        y = y | (x & 0xFF);
        x = x >> 8;
    }
    return y;
}
```

Pertama, pada fungsi dideklarasikan sebutah variable integer y yang digunakan untuk mengembalikan nilai hasil reverse. Dideklarasikan juga variable integer untuk increment looping dengan for loop untuk mengakses setiap byte dalam integer yaitu 4 byte.

Setiap putaran pengulangan diawali dengan melakukan penggeseran nilai y ke kiri (shift left) sebanyak 8 bit atau 1 byte sehingga kita dapat menambahkan byte yang baru yang telah dibalik. Kemudian untuk menambahkannya digunakan operator OR dengan x yang sudah dimasking dengan 0xFF atau bit 1 semua sehingga kita mendapatkan 1 byte pada LSB untuk dimasukkan ke variable y yang merupakan nilai yang sudah direverse. Terakhir, x geser ke kanan (shift right) 1 byte untuk menghilangkan byte yang sudah selesai dimasukkan ke variable y. Pengulangan dilakukan sebanyak 4 kali sehingga dapat melakukan reverse untuk semua byte pada data integer.

Fungsi reverseByte ini diimplementasikan dengan file main.c dengan melakukan include file header yang sudah dibuat yang berisi fungsi reverseByte. Pada file main ini juga digunakan fungsi printbitbyte pada percobaan sebelumnya untuk dapat melihat hasil dalam byte yang sudah direverse. Berikut ini adalah kode file main.c:

```
#include "reverseByte.h"
#include "printbitbyte/printbitbyte.h"

int main(){
    int x;
    printf("Masukkan x: ");
    scanf("%x", &x);
    int y = reverseBytes(x);
    printf("Hasil reverseByte(x) = 0x%x =
", y);
    printBit(sizeof(int), &y);
}
```

Pada file C ini user diminta untuk memasukkan nilai yang akan direverse berupa integer melalui terminal. Selanjutnya nilai yang dimasukkan user dijadikan parameter pada saat memanggil fungsi, hasil return fungsi ini dimasukkan ke varianbel integer y. Selanjutnya untuk melihat hasil reverse nilai y diprint dalam hexadecimal den dalam binary dengan fungsi printBit.

Hasil eksekusi file dan ketika dimasukkan suatu nilai menghasilkan reverse byte yang benar, termasuk testcase pada modul sehingga fungsi yang dibuat dan diimplementasikan sudah benar dan bekerja dengan baik.

E. Tugas 5: Fungsi Pengurangan Byte

Percobaan 5 yaitu membuat fungsi pengurangan byte 2 data dengan sistem bilangan 2's complement. Dalam merealisasikan fungsi pengurangan ini hanya diperbolehkan untuk

menggunakan operator inverse dan penjumlahan. Berikut ini adalah deklarasi fungsi yang akan dibuat yang ada pada modul:

```
#include <stdio.h>
int minBytes (int x, int y);
```

Deklarasi fungsi dilakukan di dalam file header beserta dengan include library standar untuk melakukan input dan juga print pada terminal. Fungsi ini mengembalikan nilai berupa integer dengan dua parameter interger juga yang menjadi operan untuk operasi pengurangan.

Fungsi ini kemudian dibuat dalam file C dengan melakukan include pada file header yang sudah dibuat. Berikut ini adalah file C untuk fungsi minBytes:

```
#include "minBytes.h"

int minBytes(int x, int y) {
    int result = x + (~y + 1);
    return result;
}
```

Pada fungsi tersebut, nilai hasil pengurangan dimasukkan kedalam sebuah variable integer result. Untuk melakukan pengurangan, nilai x ditambah dengan nilai y yang di inverse kemudian ditambah 1 yang merupakan minus dari nilai y tersebut pada sistem 2's complement, sehingga, hasil yang didapatkan adalah nilai x kurang y.

Fungsi tersebut kemudian dicoba pada file main dengan melakukan include file header yang sudah dibuat sebelumnya. Pada main file juga diinclude fungsi printbitbyte pada percobaan sebelumnya untuk melihat hasil dalam bit dan byte. Berikut ini adalah kode main.c:

```
#include "minBytes.h"
#include "printbitbyte/printbitbyte.h"

int main(){
    int x, y;
    printf("Masukkan x: ");
    scanf("%x", &x);
    printf("Masukkan y: ");
    scanf("%x", &y);
    int z = minBytes(x, y);
    printf("Hasil minBytes(x, y) =
0x%x\n", z);
}
```

Pada file main, user diminta untuk memasukkan nilai untuk operan x dan y yang akan dikurangkan. Hasil input user dimasukkan kedalam fungsi pada parameter fungsinya. Selanjutnya return fungsinya dimasukkan kedalam variable integer z yang kemudian diprint ke terminal.

Hasil percobaan pada file main untuk fungsi byte ini menghasilkan hasil pengurangan yang benar termasuk test case yang diberikan pada modul sehingga fungsi yang dibuat sudah benar dan berjalan dengan baik.

F. Tugas 6: Fungsi Shift Register

Percobaan 6 adalah tugas membuat fungsi shift register. Nilai yang akan dishift berupa 0 pada awalnya yang kemudian dishift suatu nilai sebanyak 5-bit secara bergantian. Berikut ini adalah deklarasi fungsi yang harus dibuat yang diberikan di modul:

```
#include <stdio.h>
int shiftRegister (int x);
```

Deklarasi fungsi dilakukan di file header shiftRegister.h yang nantinya diinclude pada file fungsinya. Pada file header juga melakukan include standard library untuk melakukan input dan output pada terminal.

Fungsi dibuat pada file shiftRegister.c dengan memanggil file header yang sudah dibuat. Berikut ini adalah fungsi yang dibuat dalam file shiftRegister.c:

```
#include "shiftRegister.h"

int y;

int shiftRegister(int x){
    y = (y<<5 | x);
    return y;
}
```

Fungsi ini mengembalikan nilai (return) berupa integer yang merupakan hasil shift yang dilakukan. Pertama, sebuah integer y dideklarasikan untuk mengembalikan nilai hasil shift. Kemudian pada fungsi, nilai y digeser ke kiri sebanyak 5 bit dan dioperasikan dengan operator OR dengan x yang merupakan bit yang akan di regist. Dengan begitu nilai 5 bit x akan masuk kedalam y, kemudian nilai yang di kembalikan (return).

Fungsi ini dicoba digunakan dalam file main.c dengan melakukan include file header fungsi ini dan juga fungsi printbitbyte yang sudah dibuat pada percobaan sebelumnya untuk melihat hasil dalam bit dan byte. Berikut ini adalah kode main.c file dalam penggunaan fungsi shiftRegister:

```
#include "shiftRegister.h"
#include "printbitbyte/printbitbyte.h"

int main(){
    int result;
    result = shiftRegister(0x04);
    printf("shiftRegister(0x04) [byte] =
0x");
    printByte((byte_pointer) &result,
sizeof(int));
    printf("shiftRegister(0x04) [bit ]
=");
    printBit(sizeof(int), &result);

    result = shiftRegister(0x13);
    printf("shiftRegister(0x13) [byte] =
0x");
    printByte((byte_pointer) &result,
sizeof(int));
    printf("shiftRegister(0x13) [bit ]
=");
```

```
    printBit(sizeof(int), &result);
    return 0;
}
```

Pada file main ini dideklarasikan sebuah variable integer untuk memasukkan hasil shift dari fungsi Bernama result. Kemudian fungsi ini dipanggil dengan parameter nilai yang akan di regist. Hasil pengembalian fungsi diprint dengan byte dan bit.

Hasil percobaan pada file main untuk fungsi byte ini menghasilkan hasil pengurangan yang benar termasuk test case yang diberikan pada modul sehingga fungsi yang dibuat sudah benar dan berjalan dengan baik.

G. Tugas 7: Program Enkripsi Sederhana

Percobaan 7 yaitu tugas membuat fungsi enkripsi sederhana dengan operator XOR. Cara encripsi yang dilakukan adalah dengan melakukan operasi XOR untuk setiap byte data yang akan dienkripsi dengan enkriptor. Berikut ini adalah deklarasi fungsi yang akan dibuat yang diberikan pada modul:

```
#include <stdio.h>
#include "printbitbyte/printbitbyte.h"
#include "shiftRegister/shiftRegister.h"

int encrypt (int x, short encryptor);

int decrypt (int x, short encryptor);
```

Deklarasi dilakukan pada file header, selain itu pada file header juga melakukan include untuk standard library dan juga fungsi bantuan yang akan digunakan pada pembuatan fungsi ini yaitu fungsi shiftRegister dan fungsi printbitbyte yang dibuat pada percobaan sebelumnya.

Selanjutnya fungsi dibuat pada file encryptDecrypt.c dengan menginclude file header yang sudah dibuat. Berikut ini adalah file C yang dibuat:

```
#include "encryptDecrypt.h"

int encrypt (int x, short encryptor){
    int encrypted = 0;
    int temp = 0;
    int i;

    temp = temp | encryptor;
    for (i = 0; i < 4; i++){
        temp = shiftRegister(encryptor);
    }
    encrypted = x ^ temp;
    return encrypted;
}

int decrypt (int x, short encryptor){
    int decrypted = 0;
    int temp = 0;
    int i;

    temp = temp | encryptor;
    for (i = 0; i < 4; i++){
        temp = shiftRegister(encryptor);
    }
    decrypted = x ^ temp;
    return decrypted;
}
```

Pada fungsi encrypt, menerima parameter integer untuk data yang akan dienkripsi dan short untuk enkriptornya. Pada fungsi encriptor dibuat variable temp sebagai variable sementara dalam proses encripsi dan variable encrypted untuk menyimpan dengan mengembalikan nilai hasil enkripsi.

Untuk melakukan enkripsi, pertama variable temp yang bernilai 0 32 bit di OR kan dengan encriptor sehingga LSB temp bernilai encriptornya. Selanjutnya dilakukan looping dengan for sebanyak 4 kali untuk melakukan encripsi terhadap 4 byte data. Pada setiap loopnya, variable temp di shift register dengan fungsi shiftRegister pada percobaan sebelumnya yang dimodifikasi untuk dapat melakukan shift registing sebanyak 8 bit. Dengan melakukan shif registing pada temp, maka semua byte dalam variable temp akan bernilai sama dengan encriptor. Sehingga, ketika loop selesai, tinggal dilakukan enkripsi saja pada nilai yang akan dienkripsi yaitu operator XOR sehingga didapatkan hasil enkripsi.

Untuk proses dekripsi dilakukan hal yang sama dengan yang dilakukan pada saat enkripsi sehingga hasil dekripsi mendapat nilai yang sama sebelum dienkripsi.

Fungsi ini diuji untuk digunakan pada file main.c. Pada main.c fungsi digunakan dengan menginclude file header dimana fungsi dideklarasikan. Berikut adalah file main.c:

```
#include "encryptDecrypt.h"

short encryptor = 85;
int x = 123456789;

int main(){
    printf("masukkan nilai x: ");
    scanf("%d", &x);

    printf("nilai x: %d\n", x);
    printf("nilai x dalam bit: ");
    printBit(sizeof(int), &x);

    int encrypted = encrypt(x,
encryptor);
    printf("nilai x setelah dienkripsi:
%d\n", encrypted);
    printf("nilai x setelah dienkripsi
dalam bit: ");
    printBit(sizeof(int), &encrypted);

    int decrypted = decrypt(encrypted,
encryptor);
    printf("nilai x setelah didekripsi:
%d\n", decrypted);
}
```

Pada file main ini dicoba melakukan enkripsi untuk nilai yang sama dengan testcase yang diberikan pada modul dan user juga dapat melakukan input pada terminal untuk melakukan

enkripsi terhadap suatu nilai. Hasil enkripsi ditampilkan dalam bit menggunakan fungsi printbitbyte pada percobaan sebelumnya.

Hasil percobaan pada file main untuk fungsi byte ini menghasilkan hasil pengurangan yang benar termasuk test case yang diberikan pada modul sehingga fungsi yang dibuat sudah benar dan berjalan dengan baik.

H. Tugas 8: Pointer dalam Assembly

Percobaan berikutnya pada tugas 8 adalah melakukan kompilasi pada suatu fungsi yang diberikan pada modul. Pada fungsi yang diberikan pada modul terdapat pointer. File tersebut kemudian dilakukan kompilasi menjadi assembly.

I. Tugas 9: Fungsi Membalik Urutan Array

J. Tugas 10: Matriks Nama

K. Tugas 11: Matriks Nama dengan Pointer

L. Tugas 12: Perkalian Matriks

M. Tugas 13: Penjumlahan Biner dengan Array

IV. SIMPULAN

- .
- .
- .
- .

REFERENSI

[1] Bryant, Randal, dan David O'Hallaron. *Computer Systems: A Programmer's Perspective 2nd Edition*. 2011. Massachusetts: Pearson Education Inc.

[2] Patterson, David, dan John Hennessy. *Computer Organization and Design: The Hardware/Software Interface*. 2012. Waltham: Elsevier Inc.

[3] Kernighan, Brian, dan Dennis Ritchie. The C Programming Language 2nd edition. 1988. Englewood Cliffs : Prentice Hall.

Lampiran

1) *Source code untuk tugas 1*

bitXor.h

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 1
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : bitXOR.h
// Deskripsi : bit XOR

#include <stdio.h>

int bitXor (int x, int y);
```

bitXor.c

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 1
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : bitXor.c
// Deskripsi : bit XOR

#include "bitXor.h"

int bitXor(int x, int y) {
  return ~(~x & ~y) & ~(x & y);
}
```

main.c

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 1
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : main.c
// Deskripsi : bit XOR

#include "bitXor.h"
#include "printbitbyte/printbitbyte.h"

int main() {
    int x;
    int y;
    printf("Masukkan nilai x: ");
    scanf("%d", &x);
    printf("Masukkan nilai y: ");
    scanf("%d", &y);
```

```
    printf("x = %d", x); printBit(sizeof(x), &x);
    printf("y = %d", y); printBit(sizeof(y), &y);

    int z = bitXor(x, y);
    printf("z = %d", z); printBit(sizeof(z), &z);
    return 0;
}
```

makefile

```
all:
  gcc -o main main.c bitXor.c printbitbyte/printbitbyte.c
clean:
  rm main
```

2) *Source code untuk tugas 2*

getByte.h

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 2
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : getByte.h
// Deskripsi : getByte


#include <stdio.h>

int getByte (int x, int n);
```

getByte.c

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 2
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : getByte.c
// Deskripsi : getByte

#include "getByte.h"

int getByte(int x, int n){
    return (x >> (n << 3)) & 0xFF;
}
```

main.c

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 2
// Tanggal : 6 Oktober 2022
// Kelompok : 10
```

```c
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : main.c
// Deskripsi : getByte

#include "getByte.h"
#include "printbitbyte/printbitbyte.h"

int main() {
    int x;
    int n;
    printf("Masukkan nilai x: ");
    scanf("%x", &x);
    printf("Masukkan nilai n: ");
    scanf("%d", &n);

    printf("Byte x = 0x "); printByte((byte_pointer) &x, sizeof(x));

    int z = getByte(x, n);
    printf("byte ke %d = 0x%x", n, z);
    return 0;
}
```

makefile

```
all:
  gcc -o main main.c getByte.c printbitbyte/printbitbyte.c
clean:
  rm main
```

3) Source code untuk tugas 3

bitMask.h

```c
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 3
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : bitMask.h
// Deskripsi : bitMask

#include <stdio.h>

int bitMask (int highbit, int lowbit);
```

bitMask.c

```c
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 3
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : bitMask.c
// Deskripsi : bitMask
```

```c
#include "bitMask.h"

int bitMask(int highbit, int lowbit){
    int mask = 0;
    int i;
    for (i = lowbit; i <= highbit; i++){
        mask |= (1 << i);
    }
    return mask;
}
```

main.c

```c
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 3
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : main.c
// Deskripsi : bitmask

#include "bitMask.h"
#include "printbitbyte/printbitbyte.h"

int main(){
    int highbit, lowbit;
    printf("Masukkan highbit: ");
    scanf("%d", &highbit);
    printf("Masukkan lowbit: ");
    scanf("%d", &lowbit);
    int mask = bitMask(highbit, lowbit);
    printf("Hasil mask = 0x%x = ", mask);
    printBit(sizeof(int), &mask);
}
```

makefile

```
all:
  gcc -o main main.c bitMask.c printbitbyte/printbitbyte.c
clean:
  rm main
```

4) *Source code untuk tugas 4*

reverseByte.h

```c
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 4
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : reverseByte.h
// Deskripsi : reverseByte

#include <stdio.h>
```

```
int reverseBytes (int x);
```

reverseByte.c

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 4
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : reverseByte.c
// Deskripsi : ReverseByte

#include "reverseByte.h"

int reverseBytes(int x){
    int y = 0;
    int i;
    for (i = 0; i < 4; i++){
        y = y << 8;
        y = y | (x & 0xFF);
        x = x >> 8;
    }
    return y;
}
```

main.c

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 4
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : main.c
// Deskripsi : reversebyte

#include "reverseByte.h"
#include "printbitbyte/printbitbyte.h"

int main(){
    int x;
    printf("Masukkan x: ");
    scanf("%x", &x);
    int y = reverseBytes(x);
    printf("Hasil reverseByte(x) = 0x%x = ", y);
    printBit(sizeof(int), &y);
}
```

makefile

```
all:
  gcc -o main main.c reverseByte.c printbitbyte/printbitbyte.c
clean:
  rm main
```

5) *Source code untuk tugas 5*

minBytes.h

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 5
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : minBytes.h
// Deskripsi : minBytes

#include <stdio.h>
// subraction of two integers two's complement
int minBytes (int x, int y);
```

minBytes.c

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 5
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : minBytes.c
// Deskripsi : minBytes

#include "minBytes.h"

// subraction of two integers two's complement only using operator + and ~
int minBytes(int x, int y) {
    int result = x + (~y + 1);
    return result;
}
```

main.c

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 5
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : main.c
// Deskripsi : minBytes

#include "minBytes.h"
#include "printbitbyte/printbitbyte.h"

int main(){
    int x, y;
    printf("Masukkan x: ");
    scanf("%x", &x);
    printf("Masukkan y: ");
```

```
    scanf("%x", &y);
    int z = minBytes(x, y);
    printf("Hasil minBytes(x, y) = 0x%x\n", z);
}
```

makefile

```
all:
  gcc -o main main.c minBytes.c printbitbyte/printbitbyte.c
clean:
  rm main
```

6) *Source code untuk tugas 6*

shiftRegister.h

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 6
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : shiftRegister.h
// Deskripsi : shiftRegister

#include <stdio.h>

int shiftRegister (int x);
```

shiftRegister.c

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 6
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : shiftRegister.c
// Deskripsi : shiftRegister

#include "shiftRegister.h"

int y;

int shiftRegister(int x){
    y = (y<<5 | x);
    return y;
}
```

main.c

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 6
```

```c
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : main.c
// Deskripsi : shiftRegister

#include "shiftRegister.h"
#include "printbitbyte/printbitbyte.h"

int main(){
    int result;
    result = shiftRegister(0x04);
    printf("shiftRegister(0x04) [byte] = 0x");
    printByte((byte_pointer) &result, sizeof(int));
    printf("shiftRegister(0x04) [bit ] =");
    printBit(sizeof(int), &result);

    result = shiftRegister(0x13);
    printf("shiftRegister(0x13) [byte] = 0x");
    printByte((byte_pointer) &result, sizeof(int));
    printf("shiftRegister(0x13) [bit ] =");
    printBit(sizeof(int), &result);
    return 0;
}
```

makefile

```
all:
  gcc -o main main.c shiftRegister.c printbitbyte/printbitbyte.c
clean:
  rm main
```

7) *Source code untuk tugas 7*

encryptDecrypt.h

```c
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 7
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : encryptDecrypt.h
// Deskripsi : encrypt decrypt

#include <stdio.h>
#include "printbitbyte/printbitbyte.h"
#include "shiftRegister/shiftRegister.h"

int encrypt (int x, short encryptor);

int decrypt (int x, short encryptor);
```

encryptDecrypt.c

```c
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
```

```c
// Percobaan : 7
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : encryptDecrypt.h
// Deskripsi : encrypt decrypt

#include "encryptDecrypt.h"

int encrypt (int x, short encryptor){
    int encrypted = 0;
    int temp = 0;
    int i;

    temp = temp | encryptor;
    for (i = 0; i < 4; i++){
        temp = shiftRegister(encryptor);
    }
    encrypted = x ^ temp;
    return encrypted;
}

int decrypt (int x, short encryptor){
    int decrypted = 0;
    int temp = 0;
    int i;

    temp = temp | encryptor;
    for (i = 0; i < 4; i++){
        temp = shiftRegister(encryptor);
    }
    decrypted = x ^ temp;
    return decrypted;
}
```

main.c

```c
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 7
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : main.c
// Deskripsi : encrypt decrypt

#include "encryptDecrypt.h"

short encryptor = 85;
int x = 123456789;

int main(){
    printf("masukkan nilai x: ");
    scanf("%d", &x);

    printf("nilai x: %d\n", x);
    printf("nilai x dalam bit: ");
    printBit(sizeof(int), &x);

    int encrypted = encrypt(x, encryptor);
```

```
    printf("nilai x setelah dienkripsi: %d\n", encrypted);
    printf("nilai x setelah dienkripsi dalam bit: ");
    printBit(sizeof(int), &encrypted);

    int decrypted = decrypt(encrypted, encryptor);
    printf("nilai x setelah didekripsi: %d\n", decrypted);
}
```

makefile

```
all:
  gcc -o main main.c encryptDecrypt.c printbitbyte/printbitbyte.c
shiftRegister/shiftRegister.c
clean:
  rm main
```

8) *Source code untuk tugas 8*

coba.c

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 8
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : coba.c
// Deskripsi : pointer dalam assembly

void coba(int* x, int* y, int* z){
    // Kamus
    int a;
    int b;
    int c;
    int d;
    // Algoritma
    a = *x;
    b = *y;
    c = *z;
    d = a+b;
    *y = d;
    *z = b;
    *x = c;
}
```

coba.s

```
  .file"coba.c"
  .text
  .globl coba
  .def coba;.scl 2;  .type32; .endef
  .seh_proc coba
coba:
  pushq%rbp
  .seh_pushreg  %rbp
  movq %rsp, %rbp
  .seh_setframe %rbp, 0
  subq $16, %rsp
  .seh_stackalloc16
```

```
  .seh_endprologue
  movq %rcx, 16(%rbp)
  movq %rdx, 24(%rbp)
  movq %r8, 32(%rbp)
  movq 16(%rbp), %rax
  movl (%rax), %eax
  movl %eax, -4(%rbp)
  movq 24(%rbp), %rax
  movl (%rax), %eax
  movl %eax, -8(%rbp)
  movq 32(%rbp), %rax
  movl (%rax), %eax
  movl %eax, -12(%rbp)
  movl -4(%rbp), %edx
  movl -8(%rbp), %eax
  addl %edx, %eax
  movl %eax, -16(%rbp)
  movq 24(%rbp), %rax
  movl -16(%rbp), %edx
  movl %edx, (%rax)
  movq 32(%rbp), %rax
  movl -8(%rbp), %edx
  movl %edx, (%rax)
  movq 16(%rbp), %rax
  movl -12(%rbp), %edx
  movl %edx, (%rax)
  nop
  addq $16, %rsp
  popq %rbp
  ret
  .seh_endproc
  .ident "GCC: (Rev9, Built by MSYS2 project) 11.2.0"

    temp = temp | encryptor;
    for (i = 0; i < 4; i++){
        temp = shiftRegister(encryptor);
    }
    decrypted = x ^ temp;
    return decrypted;
}
```

coba_double.c

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 8
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : coba_double.c
// Deskripsi : pointer dalam assembly

void coba(int* x, int* y, int* z){
    // Kamus
    double a;
    double b;
    double c;
    double d;
    // Algoritma
    a = *x;
    b = *y;
    c = *z;
```

```
    d = a+b;
    *y = d;
    *z = b;
    *x = c;
}
```

coba_double.s

```
  .file"coba_double.c"
  .text
  .globl coba
  .def coba;.scl 2;  .type32; .endef
  .seh_proc coba
coba:
  pushq%rbp
  .seh_pushreg  %rbp
  movq %rsp, %rbp
  .seh_setframe %rbp, 0
  subq $32, %rsp
  .seh_stackalloc32
  .seh_endprologue
  movq %rcx, 16(%rbp)
  movq %rdx, 24(%rbp)
  movq %r8, 32(%rbp)
  movq 16(%rbp), %rax
  movl (%rax), %eax
  pxor %xmm0, %xmm0
  cvtsi2sdl %eax, %xmm0
  movsd%xmm0, -8(%rbp)
  movq 24(%rbp), %rax
  movl (%rax), %eax
  pxor %xmm0, %xmm0
  cvtsi2sdl %eax, %xmm0
  movsd%xmm0, -16(%rbp)
  movq 32(%rbp), %rax
  movl (%rax), %eax
  pxor %xmm0, %xmm0
  cvtsi2sdl %eax, %xmm0
  movsd%xmm0, -24(%rbp)
  movsd-8(%rbp), %xmm0
  addsd-16(%rbp), %xmm0
  movsd%xmm0, -32(%rbp)
  movsd-32(%rbp), %xmm0
  cvttsd2sil%xmm0, %edx
  movq 24(%rbp), %rax
  movl %edx, (%rax)
  movsd-16(%rbp), %xmm0
  cvttsd2sil%xmm0, %edx
  movq 32(%rbp), %rax
  movl %edx, (%rax)
  movsd-24(%rbp), %xmm0
  cvttsd2sil%xmm0, %edx
  movq 16(%rbp), %rax
  movl %edx, (%rax)
  nop
  addq $32, %rsp
  popq %rbp
  ret
  .seh_endproc
  .ident "GCC: (Rev9, Built by MSYS2 project) 11.2.0"
```

9) *Source code untuk tugas 9*

main.c

```c
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 9
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : main.c
// Deskripsi : Fungsi Membalik Urutan Array

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

int main(){
    char input[100];
    int i;

    printf("Masukkan kalimat: ");
    fgets(input, 100, stdin);
    for (i = strlen(input); i >= 0; i--){
        printf("%c", input[i]);
    }
    printf("\n");
}
```

10) Source code untuk tugas 10

main.c

```c
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 10
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : main.c
// Deskripsi : Matriks Nama

#include<stdio.h>
#include<string.h>

int main() {
     char nama_orang[100][100];
     int i = 0;

     printf("Masukkan nama orang (selesai masukkan titik) : \n");
     while (1) {
          fgets(nama_orang[i], 100, stdin);
          if (strcmp(nama_orang[i], ".\n") == 0) {
               break;
          } else {
               i++;
          }
     }

     printf("\nNama orang yang dimasukkan: \n");
     for (int j = 0; j < i; j++) {
          printf("%s", nama_orang[j]);
     }
```

```
    return 0;
}
```

11) *Source code untuk tugas 11*

main.c

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 11
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : main.c
// Deskripsi : Matriks Nama dengan Pointer

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

int main() {
    char *nama_orang[100];

    char **ptrArr = nama_orang;
    char temp[100];
    int i = 0;

    printf("Masukkan nama orang: \n");
    while (1) {
        fgets(temp, 100, stdin);
        if (strcmp(temp, ".\n") == 0) {
            break;
        } else {
            ptrArr[i] = strdup(temp);
            i++;
        }
    }

    printf("\nNama orang yang dimasukkan: \n");
    for(int j = 0; j < i; j++) {
        printf("%s", ptrArr[j]);
    }
    return 0;
}
```

12) *Source code untuk tugas 12*

main.c

```
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 12
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Gilbert Ng (13220032)
// Nama (NIM) 2 : Ahmad Aziz (13220034)
// Nama File : mulmatriks.c
// Deskripsi : program perkalian 2 matriks

#include <stdlib.h>
```

```c
#include <stdio.h>


int row1, col1, row2, col2;

int mulMatriks(int A[row1][col1], int B[row2][col2]){
    int result[row1][col2];
    for (int i = 0; i < row1; i++){
        for (int j = 0; j < col2; j++){
            result[i][j] = 0;
            for (int k = 0; k < col1; k++){
                result[i][j] += A[i][k]*B[k][j];
            }
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }
}

int main(){
    printf("Masukkan jumlah baris matriks pertama: ");
    scanf("%d", &row1);
    printf("Masukkan jumlah kolom matriks pertama: ");
    scanf("%d", &col1);
    printf("\nMasukkan jumlah baris matriks kedua: ");
    scanf("%d", &row2);
    printf("Masukkan jumlah kolom matriks kedua: ");
    scanf("%d", &col2);

    int valid;
    if (row1 == col2){
        valid = 1;
    } else {
        valid = 0;
    }

    if (!valid){
        printf("\nPerkalian matriks tidak dapat dilakukan.");
    } else{
        int matrixA[row1][col1], matrixB[row2][col2];
        //input matriks
        printf("\n");
        for (int i = 1; i < row1+1; i++){
            for (int j = 1; j < col1+1; j++){
                printf("Masukkan elemen ke-[%d][%d] matriks 1: ", i, j);
                scanf("%d", &matrixA[i-1][j-1]);
            }
        }
        printf("\n");
        for (int i = 1; i < row2+1; i++){
            for (int j = 1; j < col2+1; j++){
                printf("Masukkan elemen ke-[%d][%d] matriks 2: ", i, j);
                scanf("%d", &matrixB[i-1][j-1]);
            }
        }
        printf("Hasil perkalian matriks 1 dan matriks 2:\n");
        mulMatriks(matrixA, matrixB);
    }
}
```

13) *Source code untuk tugas 13*

main.c

```c
// Praktikum EL3111 Arsitektur Sistem Komputer
```

```c
// Modul : 2
// Percobaan : 13
// Tanggal : 6 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Gilbert Ng (13220032)
// Nama (NIM) 2 : Ahmad Aziz (13220034)
// Nama File : binerarray.c
// Deskripsi : Penjumlahan Biner dengan Array

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

int main(){
    int arr1[8], arr2[8], result[8];
    int a,b, pilihan;
    printf("1. Penjumlahan\n2. Pengurangan\nPilih operasi yang ingin dilakukan
(1/2): ");
    scanf("%d", &pilihan);

    printf("Masukkan bilangan pertama: ");
    scanf("%d", &a);
    printf("Masukkan bilangan kedua: ");
    scanf("%d", &b);

    if (pilihan == 2){
        b = ~b + 1; // pengurangan -> 2's comp
    }

    // simpan bilangan ke array
    for (int i = 0; i < 8; i++){
        arr1[i] = (a >> i) & 1;
        arr2[i] = (b >> i) & 1;
    }

    int sisa = 0;
    for (int i = 0; i < 8; i++){
        result[i] = arr1[i]^arr2[i]^sisa;
        sisa = (arr1[i]&arr2[i])|(arr1[i]&sisa)|(arr2[i]&sisa);
    }

    // konversi hasil biner ke desimal
    int hasil;
    for (int i = 0; i < 8; i++){
        hasil += result[i] * pow(2,i);
    }
    hasil -= 1;
    printf("Hasil operasi: %d", hasil);
}
```

14) *Source code untuk fungsi printbitbyte*

printbitbyte.h

```c
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 0
// Tanggal : 5 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
```

```h
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : printbitbyte.h
// Deskripsi : Demonstrasi Pointer

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

typedef unsigned char *byte_pointer;

// address = alamat dari variabel dalam memory
// size = ukuran variabel dalam memory (sizeof)
void printByte(byte_pointer address, int size);

void printBit(size_t const size, void const * const address);
```

printbitbyte.c

```c
// Praktikum EL3111 Arsitektur Sistem Komputer
// Modul : 2
// Percobaan : 0
// Tanggal : 5 Oktober 2022
// Kelompok : 10
// Rombongan : B
// Nama (NIM) 1 : Ahmad Aziz (13220034)
// Nama (NIM) 2 : Gilbert Ng (13220032)
// Nama File : printbitbyte.c
// Deskripsi : Demonstrasi Pointer

#include "printbitbyte.h"

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

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("");
}
```