Program Studi Teknik Elektro ITB

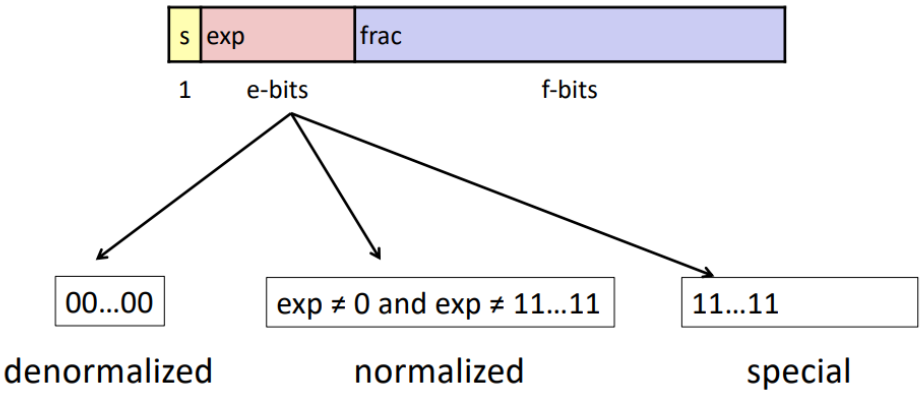
Nama Kuliah (Kode) : Praktikum Arsitektur Sistem Komputer (EL3111)
Tahun / Semester : 2022-2023 / Ganjil
Modul : POINTER, STRUCTURE, ARRAY, DAN OPERASI DALAM LEVEL BIT
Nama Asisten / NIM : _____
Nama Praktikan / NIM : Ahmad Aziz / 13220034

Tugas Pendahuluan

1. Bagaimana tipe data float dan double disimpan dalam memory komputer? Berapakah rentang (nilai minimum dan nilai maksimum) dari tipe data float dan double di luar NaN dan Inf?

Tipe data float dan double merupakan representasi bilangan "berkoma" atau bilangan diantara dua integer di dalam computer. Bilangan di dalam computer direpresentasikan dalam bentuk binary dan diterjemahkan berdasarkan tipe data bilangan tersebut. Tipe data float dan double merupakan tipe data yang sama namun dengan tingkat presisi yang berbeda, dimana tipe data float direpresentasikan dalam 32 bit yang disebut single precision dan double dengan 64 bit yang disebut double precision.

Tipe data float dan double disimpan dalam computer dengan standar tertentu agar dapat berjalan di berbagai platform. Standar tipe data float yang sering digunakan dan digunakan dalam bahasa C adalah standar IEEE 754. Pada standar IEEE 754 tipe data float disimpan dengan format berikut:



Gambar 1. IEEE 754 standard float representasion

Pada single precision 1 bit MSB digunakan sebagai sign untuk menentukan positif dan negative, dimana 0 untuk positif dan 1 untuk negative. Bit selanjutnya yaitu exponent pada single precision berjulah 8 bit. Dan bit terakhir pada LSB adalah bit fraction yang pada single precision berjumlah 23 bit.

Representasi bit tersebut mengikuti rumusan berikut pada bilangan decimal:

$$v = (-1)^s M 2^E$$

dimana, M merupakan mantissa yang berada pada rentang 1.0-2.0 dari perhitungan fraction bilangan tersebut. Sedangkan E merupakan exponent dari bilangan basis 2 yang mengikuti persamaan berikut:

$$E = exp - Bias$$

Rentang nilai tidak termasuk NaN dan infinity untuk float adalah $\pm 2^{128} - 2^{128-24}$ dan untuk double adalah $\pm 2^{1024} - 2^{1024-53}$.

2. Seperti yang kita ketahui bahwa terdapat dua jenis operator right shift, yaitu logical right shift dan arithmetic right shift. Kedua jenis operator right shift ini memiliki simbol yang sama yaitu >>. Bagaimana caranya kita dapat menentukan apakah operator right shift yang digunakan adalah logical right shift atau arithmetic right shift? Berilah contohnya dalam bentuk sintaks bahasa C!

Shift merupakan operator bitwise dimana opetor tersebut mengoperasikan data pada level bit. Dalam bahasa C terdapat operator shift left dan right dimana dapat dilakukan secara logical dan aritmatik. Untuk operator shift left, operasi secara logical maupun aritmatik akan menghasilkan output yang sama. Pada operator shift right, terdapat perbedaan output ketika dioperasikan secara logika atau aritmatik.

Cara menentukan kedua jenis operasi logika dan aritmatik pada shift right adalah dengan melihat tipe data operan. Dalam bahasa C, operan yang berbentuk singned atau yang bisa bernilai negative atau positif akan dioperasikan pada shift right secara aritmatik, sedangkan untuk operan yang berbentuk unsign atau yang hanya bernilai positif akan dioperasikan pada shift right secara logical.

Contohnya pada kode program berikut dalam bahasa C:

```c
#include <stdio.h>

int main(){
    int a = 10;
    int b = -10;
    unsigned c = 10;

    printf("a = %d\n", a>>1);    //arithmetical right shift
    printf("b = %d\n", b>>1);    //arithmetical right shift
    printf("c = %d ", c>>1);     //logical right shift

    return 0;
}
```

3. Diberikan dua buah deklarasi structure dengan elemen-elemen yang sama sebagai berikut.

```c
typedef struct{
    char kelas;
    short kode_matakuliah;
    int nim;
    char nilai_abjad;
    int nilai_angka;
} daftar_NA_1;

typedef struct{
    char kelas;
    char nilai_abjad;
    short kode_matakuliah;
    int nim;
    int nilai_angka;
} daftar_NA_2;
```

a. Berapakah ukuran structure daftar_NA_1 dan daftar_NA_2 dalam memory? Gambarkan pula bagaimana kedua structure ini disimpan dalam memory!

Ukuran kedua struktur berbeda, untuk struktur daftar_NA_1 memiliki ukuran 16 byte sedangkan struktur daftar_NA_2 memiliki ukuran 12 byte. Struktur tersebut disimpan di dalam memori dengan alokasi kontinu untuk setiap variable di dalamnya dimana dialokasikan secara berurutan sesuai dengan urutan deklarasinya.

Tabel 1. Gambaran urutan dan ukuran alokasi memori strukur daftar_NA_1

kelas	kode_matakuliah	NIM	Nilai_abjad	Nilai_angka
1 byte	2 byte	4 byte	1 byte	4 byte

Tabel 2. Gambaran urutan dan ukuran alokasi memori strukur daftar_NA_2

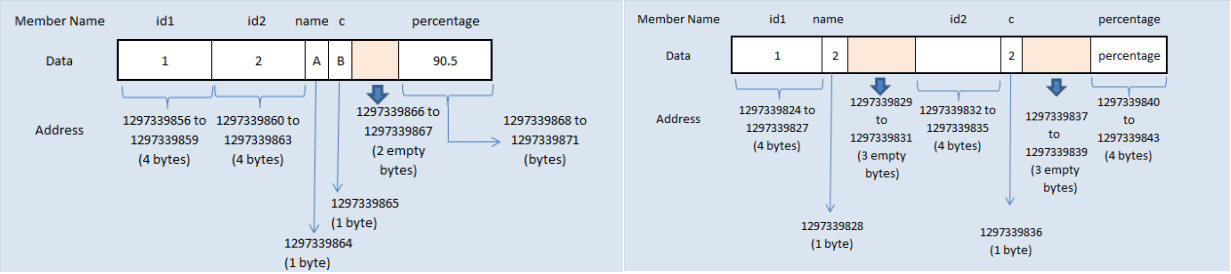
kelas	Nilai_abjad	kode_matakuliah	NIM	Nilai_angka
1 byte	1 byte	2 byte	4 byte	4 byte

b. Mengapa daftar_NA_1 dan daftar_NA_2 memiliki ukuran yang berbeda walaupun elemen-elemen penyusun structure sama namun berbeda urutan penulisan?

Pada sistem alokasi memori dalam computer mengelompokkan data kedalam paket 4 bit, hal ini untuk memaksimalkan efektifitas pembacaan data pada memori karena arsitektur computer yang bekerja dengan membaca 4 byte dalam satu waktu. Oleh karena itu, akan terdapat byte yang kosong pada alokasi memori karena sistem paket 4 byte ini jika terdapat sisa byte dan data yang akan dialokasikan berikutnya tidak muat didalam paket tersebut. Konsep ini disebut padding.

Dalam kasus alokasi memori pada struktur, inilah yang menyebabkan terjadinya perbedaan pada 2 struktur yang hanya berbeda urutan deklarasinya saja. Dimana pada strukur daftar_NA_1 sedemikian rupa menyebabkan lebih banyan byte kosong pada alokasi memori struktur tersebut. Sehingga, ukuran struktur daftar_NA_1 lebih besar disbanding struktur_NA_2.

Contoh kasus perbedaan urutan ini dapat diperhatikan pada gambar berikut:



Gambar 2. Padding alokasi memory

Sumber: https://fresh2refresh.com/c-programming/c-structure-padding/

4. Diketahui deklarasi array dua dimensi. Bagaimana komputer menyimpan array ini di dalam memory? Gambarkan bentuk penyimpanan array dua dimensi ini di dalam memory komputer.

Array dua dimensi bisa dianggap sebagai array satu dimensi yang elemennya berisi array juga. Alokasi memori pada computer untuk tipe data array dua dimensi juga mengikuti konsep ini. Jika kita membuat sebuah array dua dimensi dengan ukuran 3x3, maka akan terlihat seperti berikut:

Tabel 2. Array dua dimensi

A(0,0)	A(0,1)	A(0,2)
A(1,0)	A(1,1)	A(1,2)
A(2,0)	A(2,1)	A(2,2)

Maka alokasi memori untuk array tersebut dapat digambarkan pada table gambaran alokasi memori berikut ini:

A(0,0)	A(0,1)	A(0,2)	A(1,0)	A(1,1)	…	A(2,1)	A(2,2)

5. Diberikan contoh program sederhana sebagai berikut. Program ini akan digunakan pada saat praktikum sehingga disarankan praktikan telah menyalin source code program ini.

```
#include "printbitbyte.h"
#include <stdlib.h>
#include <stdio.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){
    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("");
}
```

Buatlah sebuah program sederhana yang menerima input sebuah bilangan integer lalu menampilkan representasinya baik dalam bit maupun byte menggunakan kedua fungsi tersebut. Program dapat dimodifikasi untuk tipe bilangan yang lain seperti float atau double.

Berikut ini adalah kode untuk menampilkan representasi dalam bit dan byte menggunakan fungsi yang diberikan:

Pertama, kode dimodifikasi sedikit dimana deklarasi library, tipe data dipindahkan ke file header printbitbyte.h agar kode lebih rapi

```
// 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.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);
```

Selanjutnya buat file main.c untuk memanggil fungsi printBit() dan printByte() dengan memanggil file header di atas yang berisi deklarasi fungsi, tipe data, dan linrary pada file printbitbyte.c yang diberikan. Selain itu, pada main juga ditambahkan percabangan sederhana untuk memilih tipe data yang ingin dicek.

Berikut adalah file mainnya:

```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 : main.c
// Deskripsi : Demonstrasi Pointer

#include "printbitbyte.h"

int tipe_data;

int main(){
    printf("Masukkan tipe data yang akan diuji (1 = int, 2 = float, 3 = double): ");
    scanf("%d", &tipe_data);
    if (tipe_data == 1){
        int i;
        printf("Masukkan nilai int: ");
        scanf("%d", &i);
        printf("\nNilai int byte: ");
        printByte((byte_pointer) &i, sizeof(i));
        printf("Nilai int bit: ");
        printBit(sizeof(i), &i);
    } else if (tipe_data == 2){
        float f;
        printf("Masukkan nilai float: ");
        scanf("%f", &f);
        printf("\nNilai float byte: ");
        printByte((byte_pointer) &f, sizeof(f));
        printf("Nilai float bit: ");
        printBit(sizeof(f), &f);
    } else if (tipe_data == 3){
        double d;
        printf("Masukkan nilai double: ");
        scanf("%lf", &d);
        printf("\nNilai double byte: ");
        printByte((byte_pointer) &d, sizeof(d));
        printf("Nilai double bit: ");
        printBit(sizeof(d), &d);
    } else {
        printf("Tipe data tidak valid");
    }
    return 0;
}
```