**Tugas Akhir**

**Program Aplikasi Kalkulator Sederhana**

Di susun untuk memenuhi tugas akhir Dasar Sistem Komputer

Dosen pengampu:Ali Tarmuji,S.T., M.Cs.

****

**Dasar Sistem Komputer**

**Nama:Safina Eka Anindita**

**Nim:2400018130**

**Link Github:**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS AHMAD DAHLAN**

**2024/2025**

**BAB I: PENDAHULUAN**

**1.1 Latar Belakang**

Komputasi digital memainkan peran penting dalam kehidupan modern, mulai dari aplikasi sederhana hingga kompleks. Salah satu contoh dasar penerapan logika komputasi adalah kalkulator. Kalkulator sederhana berbasis kode assembly dapat digunakan sebagai latihan pemrograman dasar pada lingkungan emulator 8086 untuk memahami pengoperasian arsitektur prosesor dan sistem input/output.

Kode assembly memberikan kontrol langsung terhadap perangkat keras dengan menggunakan instruksi tingkat rendah. Dalam pembuatan kalkulator sederhana, programmer belajar tentang pengelolaan register, manipulasi data, dan pemrograman interaksi dengan sistem operasi melalui interupsi.

**1.2 Rumusan Masalah**

1. Bagaimana cara membuat kalkulator sederhana menggunakan kode assembly pada emulator 8086?
2. Instruksi apa saja yang digunakan untuk mendukung operasi penjumlahan, pengurangan, perkalian, dan pembagian?

**1.3 Tujuan**

1. Membuat kalkulator sederhana yang mampu melakukan operasi aritmatika dasar (penjumlahan, pengurangan, perkalian, dan pembagian).

Memahami cara kerja emulator 8086 dan instruksi assembly dasar.

**BAB II: LANDASAN TEORI**

**2.1 Emulator 8086**

Emulator 8086 adalah perangkat lunak yang meniru arsitektur prosesor Intel 8086. Emulator ini memungkinkan pengguna menjalankan dan menguji kode assembly tanpa memerlukan perangkat keras fisik. Instruksi assembly pada prosesor ini terdiri dari operasi manipulasi data, kontrol alur, dan operasi aritmatika.

**2.2 Bahasa Assembly**

Bahasa assembly adalah bahasa pemrograman tingkat rendah yang digunakan untuk menulis program langsung dalam instruksi prosesor. Assembly memberikan akses langsung ke register CPU dan memori, sehingga sangat efisien tetapi memerlukan pemahaman mendalam tentang arsitektur sistem.

**2.3 Operasi Dasar pada Assembly**

1. **Penjumlahan (ADD):** Menambahkan dua nilai.
2. **Pengurangan (SUB):** Mengurangi satu nilai dengan nilai lainnya.
3. **Perkalian (MUL):** Mengalikan dua nilai.
4. **Pembagian (DIV):** Membagi dua nilai (dengan hasil dibagi menjadi bilangan bulat dan sisa).

**BAB III: METODOLOGI**

**3.1 Perancangan Sistem**

Sistem dirancang untuk menerima input dari pengguna berupa dua bilangan dan jenis operasi yang akan dilakukan. Program kemudian memproses operasi tersebut dan menampilkan hasilnya.

**3.2 Tahapan Implementasi**

1. **Inisialisasi Program**
   * Mengatur segmen data dan kode.
2. **Penerimaan Input**
   * Menggunakan fungsi INT 21h untuk membaca input pengguna.
3. **Pemrosesan Data**
   * Operasi aritmatika dilakukan berdasarkan input pilihan pengguna.
4. **Menampilkan Hasil**
   * Hasil operasi ditampilkan kembali ke layar menggunakan fungsi output.

**3.3 Alat dan Bahan**

1. PC atau laptop dengan emulator 8086 (misalnya TASM atau EMU8086).
2. Editor teks untuk menulis kode assembly.

### ****BAB IV: IMPLEMENTASI PROGRAM****

### 4.1 Kode Program

### Berikut adalah kode lengkap program kalkulator sederhana:

### 4.2 Langkah Langkah pembuatan

### 1. Desain program

### Menentukkan struktur utama pada kalkulator sederhana,menggunakan loop dan interups untuk perhitungan kalkulator sederhana

### 

### 2,penulisan kode essembly

### Memanfaatkan fungsi int 21 H untuk input/output ,menggunakan register untuk menyimpan hasil kalkulator sederhana

### 3.Debuging dan uji coba

### Menggunakan debuggerv EMU8086 untuk menguji jalannya program

### 

### 4.3 Kode Program

### org 100h ; Mulai program di alamat 100h

### ; Data

### menu db 'Kalkulator Sederhana', 0Dh, 0Ah

### db '1. Penjumlahan', 0Dh, 0Ah

### db '2. Pengurangan', 0Dh, 0Ah

### db 'Pilih operasi (1-2): $'

### input\_msg db 0Dh, 0Ah, 'Masukkan bilangan pertama: $'

### input\_msg2 db 0Dh, 0Ah, 'Masukkan bilangan kedua: $'

### result\_msg db 0Dh, 0Ah, 'Hasil: $'

### buffer db 6 dup('$') ; Buffer untuk angka yang akan ditampilkan

### .code

### start:

### ; Menampilkan menu

### mov ah, 09h

### lea dx, menu

### int 21h

### ; Meminta pengguna memilih operasi

### mov ah, 01h

### int 21h ; Membaca karakter pilihan

### sub al, '0' ; Mengonversi karakter ke angka

### mov bl, al ; Menyimpan pilihan operasi di BL

### ; Meminta input angka pertama

### lea dx, input\_msg

### mov ah, 09h

### int 21h ; Menampilkan prompt untuk angka pertama

### call input\_number

### mov cx, ax ; Menyimpan bilangan pertama di CX

### ; Meminta input angka kedua

### lea dx, input\_msg2

### mov ah, 09h

### int 21h ; Menampilkan prompt untuk angka kedua

### call input\_number

### mov dx, ax ; Menyimpan bilangan kedua di DX

### ; Menangani operasi berdasarkan pilihan

### cmp bl, 1 ; Cek apakah pilihan 1 (Penjumlahan)

### je addition

### cmp bl, 2 ; Cek apakah pilihan 2 (Pengurangan)

### je subtraction

### jmp exit

### addition:

### add cx, dx ; CX = CX + DX (Penjumlahan)

### mov ax, cx ; Menyimpan hasil penjumlahan ke AX

### call display\_result

### jmp start ; Kembali ke menu utama

### subtraction:

### sub cx, dx ; CX = CX - DX (Pengurangan)

### mov ax, cx ; Menyimpan hasil pengurangan ke AX

### call display\_result

### jmp start ; Kembali ke menu utama

### ; Fungsi untuk input bilangan

### input\_number:

### xor ax, ax ; Bersihkan AX

### mov cx, 0 ; Menyimpan hasil input

### mov bl, 10 ; Basis 10 untuk konversi

### input\_loop:

### mov ah, 01h ; Membaca karakter

### int 21h

### sub al, '0' ; Mengonversi karakter ke angka

### cmp al, 0 ; Cek apakah angka valid

### jl done\_input

### mul bl ; AX = AX \* 10

### add ax, cx ; Menambahkan angka ke hasil konversi

### mov cx, ax ; Simpan hasil konversi

### jmp input\_loop

### done\_input:

### ret

### ; Fungsi untuk menampilkan hasil

### display\_result:

### mov ah, 09h

### lea dx, result\_msg

### int 21h

### mov ax, cx

### call print\_number

### ret

### ; Fungsi untuk mencetak angka

### print\_number:

### xor bx, bx ; Bersihkan BX

### mov cx, 10 ; Basis 10

### mov si, 0 ; Indeks buffer

### convert\_loop:

### xor dx, dx

### div cx ; Membagi AX dengan 10

### add dl, '0' ; Konversi ke karakter ASCII

### push dx ; Simpan digit ke stack

### inc si ; Increment indeks

### test ax, ax ; Cek apakah sudah habis

### jnz convert\_loop

### output\_loop:

### pop dx ; Ambil digit dari stack

### mov buffer[si-1], dl

### dec si

### loop output\_loop

### mov buffer[si], '$'

### lea dx, buffer

### mov ah, 09h

### int 21h

### ret

### exit:

### mov ah, 4Ch

### int 21h

### end start

### BAB V: HASIL DAN PEMBAHASAN

### 4.4 Hasil Implementasi

### Hasil implementasi dari kode EMU kalkulator sederhana yang telah disusun adalah program yang mampu menjalankan operasi aritmatika dasar seperti penjumlahan, pengurangan, perkalian, dan pembagian.

### Ketika program dijalankan:

### Pengguna akan diberikan menu untuk memilih jenis operasi (1-4).

### Setelah memilih operasi, pengguna diminta memasukkan dua bilangan.

### Program memproses input dan menghasilkan output sesuai dengan operasi yang dipilih.

### Untuk memastikan hasilnya:

### Jika Anda menggunakan emulator seperti EMU8086, kode tersebut dapat disalin dan dijalankan dalam lingkungan emulator.

### Setelah itu, Anda dapat menguji setiap operasi untuk memastikan outputnya sesuai dengan logika yang telah diimplementasikan.

### Berikut contoh skenario hasil:

### Input: Operasi "Penjumlahan", Bilangan pertama = 5, Bilangan kedua = 3. Output: Hasil: 8.

### Input: Operasi "Pembagian", Bilangan pertama = 10, Bilangan kedua = 0. Output: Pesan kesalahan: "Kesalahan: Pembagian dengan nol!"

### Jika ada kendala saat implementasi, beri tahu saya untuk membantu debugging atau penyesuaian lebih lanjut.

### 4.5 Analisis

### Kelebihan

### Struktur sederhana dan mudah dipahami.

### Menggunakan instruksi dasar yang cocok untuk pemula dalam assembly.

### Kekurangan

### Tidak ada validasi input selain operator.

### Tidak menangani angka negatif atau desimal.

### Tidak ada mekanisme penanganan kesalahan pembagian dengan nol.

### BAB VI : SARAN DAN KESIMPULAN

### 4.6 Saran

### Saran untuk Pengembangan

### Validasi Input

### Tambahkan validasi untuk memastikan input berupa angka dan operator yang valid.

### Tangani kasus seperti pembagian dengan nol agar program tidak mengalami error.

### Mendukung Angka Negatif dan Desimal

### Perluas fungsionalitas kalkulator untuk mendukung angka negatif dan desimal.

### Gunakan algoritma konversi untuk bilangan desimal ke binary floating-point jika diperlukan.

### Looping dan Navigasi Menu

### Tambahkan opsi untuk menghitung ulang atau keluar dari program tanpa perlu menjalankan ulang emulator.

### Gunakan logika branching untuk memberikan pengalaman pengguna yang lebih baik.

### Optimasi Kinerja

### Gunakan register secara efisien untuk meminimalkan penggunaan memori.

### Tambahkan komentar lebih detail di setiap bagian kode untuk memperjelas logika program.

### Ekspansi Operasi Matematika

### Tambahkan operasi seperti eksponen, modulus, atau akar kuadrat untuk memperluas fungsi kalkulator.

### Gunakan subrutin (procedures) untuk operasi kompleks agar kode lebih modular dan terstruktur.

### Penanganan Error yang Lebih Baik

### Implementasikan logika penanganan error untuk setiap kasus potensial, seperti overflow register atau input yang tidak valid.

### 4.7 Kesimpulan

### Kesimpulan

### Kalkulator sederhana dalam bahasa assembly untuk emulator 8086 adalah program yang dirancang untuk memahami operasi dasar prosesor 8086, seperti pengolahan input, operasi aritmatika, dan menampilkan hasil menggunakan interrupt BIOS.

### Kode ini memperlihatkan bagaimana memanfaatkan instruksi assembly dasar seperti ADD, SUB, MUL, dan DIV, serta bagaimana menggunakan interrupt INT 21h untuk interaksi input/output. Kalkulator ini cocok untuk pemula karena menyederhanakan logika pemrograman dalam konteks instruksi hardware langsung.

### Namun, kalkulator ini memiliki beberapa keterbatasan, seperti kurangnya validasi input, tidak mendukung angka negatif atau desimal, dan minimnya fitur penanganan kesalahan.

### LAMPIRAN PENUGASAN

### 

### 

### 

### 

### 