

**LAPORAN RESMI
PRAKTIKUM ARSITEKTUR KOMPUTER
“MIKROPROSESOR DAN INSTRUCTION SET”**



Disusun Oleh :

Izzuddin Ahmad Afif (2421600011)

Dosen :

Mohamad Ridwan S.T., M.T.

**PROGRAM STUDI SARJANA TERAPAN TEKNOLOGI REKAYASA INTERNET
DEPARTEMEN TEKNIK ELEKTRO
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
2021/2022**

Praktikum 1

Mikroprosesor dan Instruction Set

A. Pendahuluan:

Arsitektur komputer adalah konsep perencanaan dan struktur pengoperasian dasar dari suatu sistem komputer. Arsitektur komputer ini merupakan rencana cetak-biru dan deskripsi fungsional dari kebutuhan bagian perangkat keras yang didesain (kecepatan proses dan sistem interkoneksinya). Dalam hal ini, implementasi perencanaan dari masing–masing bagian akan lebih difokuskan terutama, mengenai bagaimana CPU akan bekerja, dan mengenai cara pengaksesan data dan alamat dari dan ke memori cache, RAM, ROM, cakram keras, dll). Beberapa contoh dari arsitektur komputer ini adalah arsitektur von Neumann, CISC, RISC, blue Gene, dll. Mikroprosesor adalah sebuah chip (IC=Integrated Circuits) yang di dalamnya terkandung rangkaian ALU (Arithmetic-Logic Unit), rangkaian CU (Control Unit) dan kumpulan register register.

Mikroprosesor disebut juga dengan CPU (Central Processing Unit) yang digunakan sebagai otak/pengolah utama dalam sebuah sistem komputer. Mikroprosesor pertama yang diproduksi adalah mikroprosesor 4bit dari intel yang diberi nama Intel 4004, lalu dikembangkan menjadi Intel 4008, lalu dikembangkan lagi ,menjadi 8 bit dengan diproduksinya seri 8008 dan 8085. Agar CPU dapat bekerja sesuai dengan yang diinginkan, maka diperlukan suatu instruksi. Setiap mikroprosesor atau CPU mempunyai satu set kode instruksi (instruction set) yang spesifik, dan berbeda antara instruction set antara CPU satu dengan yang lainnya.

Selama berlangsungnya eksekusi instruksi, instruksi dibaca ke dalam register instruksi yang terdapat dalam CPU. Untuk melakukan operasi yang diperlukan, CPU harus dapat mengeluarkan data dari berbagai bidang instruksi. Opcode direpresentasikan dengan singkatan-singkatan, yang disebut mnemonik, yang mengindikasikan operasi, contohnya adalah: ADD : Add (Menambahkan) SUB : Subtract (Pengurangan) MPY : Multiply (Perkalian) DIV : Divide (Pembagian) LOAD : Muatkan data data dari memori STOR : Simpan data ke memori .

B. Percobaan

Tools:

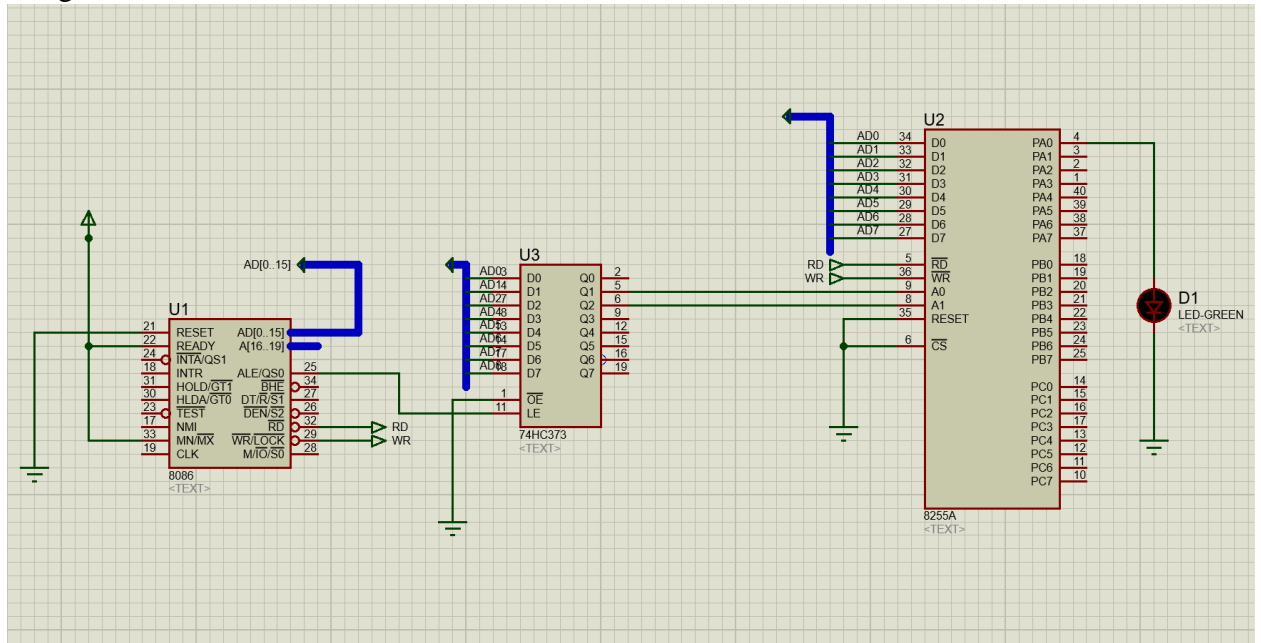
1. Proteus Professional
2. EMU8086

Bahan Percobaan:

1. Datasheet Intel 8086
2. Instruction set for Intel 8086
3. Datasheet IC 8255 PPI
4. Datasheet IC 74HC373

Hasil Percobaan:

- Rangkaian:



Source Assembly Code:

The image displays the emu8086 - assembler and microprocessor emulator 4.08 interface. The main window shows the source assembly code, which includes comments and instructions for setting up the environment, defining data, and executing a loop. A smaller window titled 'original source co...' shows a snippet of the code. The bottom window, titled 'emulator: prakarskomp1.bin', shows the registers and memory dump.

Source Assembly Code:

```
22 #SS=0500h ; same as loading segment
23 #SP=FFFEh ; set to top of loading segment
24
25 ; set general registers (optional)
26 #AX=0000h
27 #BX=0000h
28 #CX=0000h
29 #DX=0000h
30 #SI=0000h
31 #DI=0000h
32 #BP=0000h
33
34 ; add your code here
35 DATA SEGMENT
36 PORTA EQU 00h
37 PORT_COM EQU 06h
38 DATA ENDS
39
40 CODE SEGMENT
41 MOV AX, DATA
42 MOV DS, AX
43
44 org 0000h
45
46 ;add your code here
47 START:
48
49 MOV DX, PORT_COM
50 MOV AL, 10000000b; PORT A, PORT B, PORT C as output
51 OUT DX, AL
52
53 JMP XX
54
55 XX:
56 MOV AL, 0000h
57 MOV DX, PORTA
58 OUT DX, AL
59 MOV CX, 0DF36h; Delay
60 loopy1: loop loopy1
61 MOV AL, 0001h
62 MOV DX, PORTA
63 OUT DX, AL
64 MOV CX, 0DF36h; Delay
65 loopy2: loop loopy2
66 JMP XX
67
68 CODE ENDS
69 END
70
71 HLT ; halt!
72
73
74
```

original source co...

```
60 loopy1: loop loopy1
61 MOV AL, 0001h
62 MOV DX, PORTA
63 OUT DX, AL
64 MOV CX, 0DF36h; Delay
65 loopy2: loop loopy2
66 JMP XX
67
68 CODE ENDS
69 END
70
71 HLT ; halt!
72
73
74
```

emulator: prakarskomp1.bin

Registers:

Register	H	L	Value
AX	00	00	0000
BX	00	00	0000
CX	00	00	0000
DX	00	00	0000
CS	0500		0500
IP	0000		0000
SS	0500		0500
SP	FFFE		FFFE
BP	0000		0000
SI	0000		0000
DI	0000		0000
DS	0500		0500
ES	0500		0500

Memory Dump:

Address	Value
05000: B8 184	1
05001: 00 000	NULL
05002: 00 000	NULL
05003: 8E 142	2
05004: D0 216	3
05005: B0 186	4
05006: 06 006	5
05007: 00 000	NULL
05008: D0 176	6
05009: 80 128	7
0500A: EE 238	8
0500B: EB 235	9
0500C: 00 000	NULL
0500D: B0 176	10
0500E: 00 000	NULL
0500F: B0 186	11
05010: 00 000	NULL
05011: 00 000	NULL
05012: EE 238	12
05013: B9 185	13
05014: 36 054	14
05015: DF 223	15

Registers (continued):

Register	Value
MOV AX, 00000h	00000h
MOV DS, AX	00000h
MOV DX, 00006h	00006h
MOV AL, 080h	080h
OUT DX, AL	080h
MOV AL, 00h	00h
MOV DX, 00000h	00000h
OUT DX, AL	00000h
MOV CX, 0DF36h	0DF36h
LOOP 016h	016h
MOV AL, 01h	01h
MOV DX, 00000h	00000h
OUT DX, AL	00000h
MOV CX, 0DF36h	0DF36h
JMP 0Dh	0Dh
NOP	NOP
NOP	NOP
NOP	NOP
NOP	NOP

- [illegible]

- The screenshot displays a Windows desktop environment. In the background, a text editor window titled 'edit: C:\emu8086\MySource\mycode.asm' contains assembly code for an 8086 processor. The code includes register initialization, a data segment, and two code segments. A yellow highlight is present on the line 'loopy1: loop loopy1' in the second code segment. In the foreground, a command prompt window shows the execution of 'C:\emu8086\mycode.asm' and the output '00000000'. To the right, an 'emulator: tes2.bin' window is open, showing a register window with 'AX' at 0000 and 'DX' at 0000, and a memory window displaying the contents of memory addresses starting from 05003.

Analisa:

Percobaan kali membahas tentang Mikroprosesor dan set-set instruksi. Pertama kita membuat rangkaian seperti di modul, lalu kita mengetikkan code assembly seperti yang ada di modul dan dicompile dalam bentuk .bin. Lalu kita import code kita dari EMU8086 ke perangkat 8086 yang ada di rangkaian serta mengganti internal memory size nya menjadi 0x10000 dan kita run percobaannya. Setelah itu kita memindah LED pada sambungan PA7 dan mengubah program assembly supaya dihasilkan input yang diinginkan.

Di program assembly, ada kode untuk mengalirkan listrik menuju LED yang menggunakan akhiran h yang menandakan heksadesimal, di situ bila kita isi dengan 0001h maka akan menyalakan sambungan dengan label PA0 sedangkan 0080h akan menyalakan sambungan label PA7.

Kesimpulan:

Dari percobaan kali ini, dapat disimpulkan bahwa mikroprosesor dapat menerima set-set instruksi dari code assembly dan mengeksekusinya sesuai code, lalu menghasilkan sebuah input yang sesuai dengan code juga.

TUGAS:

1. Register adalah merupakan memori pada bagian komputer (mikroprocessor) yang berkapasitas kecil namun memiliki kecepatan baca yang sangat tinggi. Register pada mikroprocessor dapat diibaratkan sebagai kaki dan tangan dari mikroprocessor, karena dalam setiap pekerjaan, mikroprocessor selalu mengandalkan dan melibatkan register sebagai perantara dan sebagai komponen yang paling banyak melakukan pekerjaan mikroprocessor. Secara umum, fungsi sebuah register pada mikroprocessor adalah sebagai tempat penyimpanan sementara untuk menyimpan hasil dari operasi aritmatika ataupun operasi logika yang dilakukan oleh mikroprocessor.

2. - Register AX (AH + AL) / Accumulator Register

Register AX adalah register yang berfungsi untuk menyimpan dan membaca data yang berhubungan dengan operasi aritmatika, yang meliputi perkalian, pembagian, penjumlahan dan pengurangan (KABATAKU). Karena setiap general purpose register memiliki register High dan Low, maka untuk Register AX, register Low nya adalah AL dan Highnya adalah AH.

- **Register CX (CH + CL) / Counter Register**

Register CX merupakan memori yang berfungsi untuk menyimpan jumlah lompatan pada loop yang dilakukan oleh mikroprocessor. Secara umum, terdapat beberapa fungsi dari register CX, yaitu:

- Melakukan operasi pencacahan untuk operasi loop
- Melakukan operasi pencacahan untuk operasi shift dan rotate
- Melakukan operasi pencacahan untuk operasi string

3. Assembly Language

- a. MOV is an X86 assembly language instruction, it is meant to move data between registers and memory.
- b. JMP instruction performs an unconditional jump. Such an instruction transfers the flow of execution by changing the instruction pointer register.
- c. OUT instruction transfers a byte, word, or long from the AL, AX, or EAX registers respectively to a port (0 to 65535), specified by the DX register.