Lab 4 Clement Samuel Marly 2206082114

1. Program sederhana

Graphical user interface, application, Word

Description automatically generated

Alur program dapat dilihat di *comment lines* dalam gambar.

Program diatas menggunakan ldi, add, sub tujuan untuk melakukan operasi tambah dan kurang dalam register.

Program juga menggunakan st dan clr yang berfungsi untuk menyimpan atau menghapus value dalam register.

1. Jelaskan alur program

.include "m8515def.inc"

.def VAL = R16 set R16 sebagai VAL

.def ODD = R17 set R17 sebagai ODD

.def EVEN = R19 set R19 sebagai EVEN

ldi R16, 5 set R16 = 5

MAIN:

cpi VAL, 0 mengecek nilai VAL apabila 0 akan masuk ke breq

breq forever masuk ke dalam forever apabila VAL = 0

mov r20, VAL R20 diset dengan nilai VAL

andi r20, 1 R20 di modulus 2 kemudian dicek apakah nilainya 1 atau tidak

breq GENAP Apabila nilai 0, pindah ke GENAP

brne GANJIL Apabila nilai 1, pindah ke GANJIL

GANJIL:

add ODD, VAL menambahkan nilai ODD dengan VAL kemudian disimpan di ODD (R17)

dec VAL mengurangi nilai VAL sebanyak 1

rjmp MAIN kembali ke MAIN

GENAP: add EVEN, VAL menambahkan nilai EVEN dengan VAL kemudian disimpan di EVEN (R19)

dec VAL mengurangi nilai VAL sebanyak 1

rjmp MAIN kembali ke MAIN

forever:

rjmp forever infinite loop atau tutup program

Graphical user interface, application

Description automatically generated

Program berfungsi untuk mendeteksi bilangan ganjil dan genap kemudian menambahkan bilangan – bilangan yang ganjil ke dalam suatu register (R17) dan bilangan – bilangan yang genap ke dalam suatu register lain (R19). Pada awal program akan set VAL atau value awal 5 sehingga 5 akan dicek genap atau ganjil kemudian ditambahkan ke register yang sesuai. Setelah itu, VAL akan dikurangi 1 setiap kali selesai menambahkan bilangan genap atau ganjil ke registernya.  
Pada akhir program, R17 atau kumpulan bilangan ganjil akan berjumlah 5+3+1 atau 9 dan kumpulan bilangan genap atau R19 akan berisi 4+2 atau 6.

1. Program kantin Mang Udin

Program dan penjelasannya

.include "m8515def.inc"

//set variabel sesuai soal dan register

.def RB = R3

.def SK = R4

.def CC = R5

.def ED = R6

.def ANTRE = R7

.def TPA = R8

.def TPB = R9

.def TPC = R10

.def TPD = R11

.def cek = R17

.def count = R18

.def temporary = R19

START:

//inisiasi stack

ldi R16,low(RAMEND)

out SPL,R16

ldi R16,high(RAMEND)

out SPH,R16

Set\_harga:

//set harga

ldi R16, 7 //set R16 = 7

mov RB, R16 // copy nilai R16 ke RB (R3)

ldi R16, 5 //set R16 = 5

mov SK, R16 // copy nilai R16 ke SK (R4)

ldi R16, 5 //set R16 = 5

mov CC, R16 // copy nilai R16 ke CC (R5)

ldi R16, 4 //set R16 = 4

mov ED, R16 // copy nilai R16 ke ED (R6)

Main:

//set awal

ldi R16, 10 //set nilai antrian 10 (2114%8 + 8)

mov ANTRE, R16 // copy nilai R16 ke ANTRE (R7)

rjmp Antrian // lompat ke Antrian

Check:

//cek urutan apakah ganjil atau genap

inc count //jumlah count ditambah 1

mov cek, count //copy nilai count ke cek

mov temporary, count //copy nilai count ke temporary

andi cek, 1 //cek dimodulus 2 kemudian dicek apakah 0 atau 1 (Validasi genap atau ganjil)

breq Kelipatan\_enam //jika cek = 0 atau genap

brne Kelipatan\_tiga //jika cek = 1 atau ganjil

Kelipatan\_tiga:

//menyesuaikan urutan dengan paket yang sesuai

cpi temporary, 3 //Mengecek apakah nilai temporary sama dengan 3

breq Paket\_B //jika temporary sama dengan 3, masuk ke PAKET\_B

brlo Paket\_D //jika temporary tidak sama dengan 3, masuk ke PAKET\_D

subi temporary, 3 //temporary dikurang 3 sampai sisa

rjmp Kelipatan\_tiga //loop

Kelipatan\_enam:

//menyesuaikan urutan dengan paket yang sesuai

cpi temporary, 6 //Mengecek apakah nilai temporary sama dengan 6

breq Paket\_C //jika temporary sama dengan 3, masuk ke PAKET\_B

brlo Paket\_A //jika temporary tidak sama dengan 3, masuk ke PAKET\_D

subi temporary, 6 //temporary dikurang 3 sampai sisa

rjmp Kelipatan\_enam //loop

//masukkan nilai

Paket\_A:

mov R16, RB //copy nilai RB ke R16

add R16, CC //tambah RB dengan CC di R16

push R16 //push nilai R16 ke stack

rjmp Antrian

Paket\_B:

mov R16, SK //copy nilai SK ke R16

add R16, ED //tambah SK dengan ED di R16

push R16 //push nilai R16 ke stack

rjmp Antrian

Paket\_C:

mov R16, RB //copy nilai RB ke R16

add R16, ED //tambah RB dengan ED di R16

push R16 //push nilai R16 ke stack

rjmp Antrian

Paket\_D:

mov R16, SK //copy nilai SK ke R16

add R16, CC //tambah SK dengan ED di R16

push R16 //push nilai R16 ke stack

rjmp Antrian

//ulang loop apabila jumlah yang telah dihitung belum sesuai dengan jumlah antrian

Antrian:

cp count, ANTRE //Mengecek nilai count dengan ANTRE

brne Check //Jika nilai count tidak sama dengan ANTRE, masuk ke Check

breq Calculate //jika nilai count sama dengan ANTRE, masuk ke Calculate

//cek harga paket

Cek\_hapus:

mov cek, count //copy nilai count ke check

mov temporary, count ///copy nilai count ke temporary

andi cek, 1 //validasi genap atau ganjil

breq Cek\_enam //jika genap, cek apakah kelipatan 6

brne Cek\_tiga //jika ganjil, cek apakah kelipatan 3

//validasi tipe paket

Cek\_tiga:

cpi temporary, 3 //Mengecek apakah nilai temporary sama dengan 3 (kelipatan 3)

breq Tambah\_Paket\_B //jika iya, masuk ke Tambah\_Paket\_B

brlo Tambah\_Paket\_D //jika tidak, masuk ke Tambah\_Paket\_D

subi temporary, 3 //temporary dikurang 3 sampai sisa

rjmp Cek\_tiga //loop

Cek\_enam:

cpi temporary, 6 //Mengecek apakah nilai temporary sama dengan 6 (kelipatan 6)

breq Tambah\_Paket\_C ///jika iya, masuk ke Tambah\_Paket\_C

brlo Tambah\_Paket\_A //jika tidak, masuk ke Tambah\_Paket\_A

subi temporary, 6 //temporary dikurang 6 sampai sisa

rjmp Cek\_enam //loop

//masukkan nilai paket ke dalam register yang sesuai

Tambah\_Paket\_A:

pop R16 //ambil value R16

add TPA, R16 //tambah TPA dan R16 di TPA

dec count //decrement count

rjmp Calculate //masuk kembali ke Calculate

Tambah\_Paket\_B:

pop R16 //ambil value R16

add TPB, R16 //tambah TPB dan R16 di TPB

dec count //count dikurang 1

rjmp Calculate //masuk kembali ke Calculate

Tambah\_Paket\_C:

pop R16 //ambil value R16

add TPC, R16 //tambah TPC dan R16 di TPC

dec count //count dikurang 1

rjmp Calculate //masuk kembali ke Calculate

Tambah\_Paket\_D:

pop R16 //ambil value R16

add TPD, R16 //tambah TPD dan R16 di TPD

dec count ///count dikurang 1

rjmp Calculate //masuk kembali ke Calculate

//apabila antrian sudah habis

Calculate:

cpi count, 0 //Mengecek apakah nilai count sama dengan 0

breq forever //jika iya, program diakhiri

brne Cek\_hapus //jika tidak, masuk ke Cek\_hapus

forever:

rjmp forever // akhir program

Text

Description automatically generated

Hasil akhir =

R8 atau total paket A = 48 (30 heksadesimal)

R9 atau total paket B = 18 (12 heksadesimal)

R10 atau total paket C = 11 (0B heksadesimal)

R11 atau total paket D = 30 (1E heksadesimal)