

LAPORAN PRAKTIKUM

BAHASARAKITAN



Disusun Oleh :

Nama : Indah Cahya Resti

NIM : 09011281823046

Jurusan : Sistem Komputer

Dosen : Aditya P. P. Prasetyo, S. Kom., M.T.

LABORATORIUM ROBOTIKA DAN SISTEM KENDALI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SRIWIJAYA
INDRALAYA 2020

PRAKTIKUM III

OPERASI ARITMATIKA (LANJUTAN)

A. TUJUAN

- Mengerti dan memahami prinsip dasar dari proses operasi aritmatika
- Mengerti dan memahami setiap instruksi yang digunakan dalam proses operasi aritmatika
- Mampu menganalisa setiap data yang dihasilkan

B. PERALATAN YANG DIGUNAKAN

- Seperangkat PC
- Software *Command Prompt*
- Modul *Praktikum Bahasa Rakitan*

C. PROSEDURE PRAKTIKUM

- Salinlah program 3
- Amatilah output yang ditampilkan
- Buat Algoritma program
- Buat Flowchart program
- Buat Analisis program
- Buat Kesimpulan

D. DASAR TEORI

OPERASI PERKALIAN

Untuk perkalian bisa digunakan perintah MUL dengan syntax:

MUL Sumber

Sumber disini dapat berupa suatu register 8 bit(Mis:BL,BH,...), register 16 bit(Mis: BX,DX,..) atau suatu varibel.

Ada 2 kemungkinan yang akan terjadi pada perintah MUL ini sesuai dengan jenis perkalian 8 bit atau 16 bit. Bila Sumber merupakan 8 bit seperti **MUL BH** maka komputer akan mengambil nilai yang terdapat pada BH dan nilai pada AL untuk dikalikan. Hasil yang didapat akan selalu disimpan pada register AX. Bila sumber merupakan 16 bit seperti **MUL BX** maka komputer akan mengambil nilai yang terdapat pada BX dan nilai pada AX untuk dikalikan. Hasil yang didapat akan disimpan pada register DX dan AX(DX:AX), jadi register DX menyimpan Word tingginya dan AX menyimpan Word rendahnya. Sebelum Anda bisa mengalikan dua digit ASCII, Anda harus menutup 4 bit angka atas dari masing-masing digit. Hal ini menyebabkan *hasil* BCD (satu digit BCD per *byte*) pada setiap *byte*. Setelah dua *hasil* digit BCD dikalikan, instruksi AAM digunakan untuk mengatur hasil dari dua *hasil* digit BCD dalam AX. AAM berfungsi hanya setelah pengalihan dari dua *hasil byte* BCD, dan AAM hanya berfungsi hanya pada suatu *operand* dalam AL. AAM memperbaharui PF, SF, dan ZF, tetapi AF, CF, dan OF tidak diterangkan.

CONTOH:

```
                ; AL = 00000101 = hasil BCD 5
                ; BH = 00001001 = hasil BCD 9
MUL BH          ; AL x BH ; hasil dalam AX
                ; AX = 00000000 00101101 = 002DH
AAM             ; AX = 00000100 00000101 = 0405H,
                ; yang merupakan hasil BCD untuk angka 45.
```

; Jika menginginkan kode ASCII untuk hasilnya, gunakan instruksi berikutnya

OR AX, 3030H ; Letakkan 3 pada angka atas di setiap *byte*.

; AX = 00110100 00110101 = 3435H,

; yang merupakan kode ASCII untuk angka 45

PEMBAGIAN

Operasi pada pembagian pada dasarnya sama dengan perkalian. Untuk operasi pembagian digunakan perintah DIV dengan syntax:

DIV Sumber

Bila **sumber** merupakan operand 8 bit seperti **DIV BH**, maka computer akan mengambil nilai pada register AX dan membaginya dengan nilai BH. Hasil pembagian 8 bit ini akan disimpan pada register AL dan sisa dari pembagian akan disimpan pada register AH. Bila **sumber** merupakan operand 16 bit seperti **DIV BX**, maka komputer akan mengambil nilai yang terdapat pada register DX:AX dan membaginya dengan nilai BX. Hasil pembagian 16 bit ini akan disimpan pada register AX dan sisa dari pembagian akan disimpan pada register DX. AAD mengubah dua *hasil* digit BCD menjadi AH dan AL menjadi angka biner yang seimbang dalam AL. Pengaturan ini harus dibuat sebelum membagi dua *hasil* digit BCD dalam AX dengan *byte hasil* BCD. Setelah pembagian, AL akan berisi hasil bagi dari *hasil* BCD dan AH akan berisi sisa *hasil* BCD. PF, SF, dan ZF diperbaharui. AF, CF, dan OF tidak diterangkan setelah AAD.

CONTOH:

; AX = 0607H *hasil* BCD untuk 67 desimal

; CH = 09H, sekarang atur menjadi biner

AAD ; Hasil: AX = 0043 = 43H = 67 desimal

DIV CH ; Bagi AX dengan *hasil* BCD pada CH

; Hasil Bagi : AL = 07 *hasil* BCD

; Sisa : AH = 04 *hasil* BCD

; *Flags* tidak diterangkan setelah DIV

E. PROGRAM

Program 3

```
MOV AL,12
MOV BL,09
MOV CL,33
MOV DL,44

MUL BH
CMP AX,BX
MOV BX,0034
MOV CX,2323
MUL BX
CLC
MUL CX
CMC
MUL DX
MOV AX,9F98
DIV DX
MOV AX,FF67
MOV CX,0220
DIV CX
```

F. OUTPUT PROGRAM

TAMPILKAN OUTPUT SCREEN

```
DOSBox Status Window
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
Z:\>E:
E:\>DEBUG
-A100
072A:0100 MOV AL,12
072A:0102 MOV BL,09
072A:0104 MOV CL,33
072A:0106 MOV DL,44
072A:0108 MUL BH
072A:010A CMP AX,BX
072A:010C MOV BX,0034
072A:010F MOV CX,2323
072A:0112 MUL BX
072A:0114 CLC
072A:0115 MUL CX
072A:0117 CMC
072A:0118 MUL DX
072A:011A MOV AX,9F9B
072A:011D DIV DX
072A:011F MOV AX,FF67
072A:0122 MOV CX,0220
072A:0125 DIV CX
072A:0127
```

```
DOSBox Status Window
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
-T
AX=0012 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0102 NU UP EI NG NZ NA PO NC
072A:0102 B309 MOV BL,09
-T
AX=0012 BX=0009 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0104 NU UP EI NG NZ NA PO NC
072A:0104 B133 MOV CL,33
-T
AX=0012 BX=0009 CX=0033 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0106 NU UP EI NG NZ NA PO NC
072A:0106 B244 MOV DL,44
-T
AX=0012 BX=0009 CX=0033 DX=0044 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0108 NU UP EI NG NZ NA PO NC
072A:0108 F6E7 MUL BH
-T
AX=0000 BX=0009 CX=0033 DX=0044 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=010A NU UP EI NG NZ NA PO NC
072A:010A 39DB CMP AX,BX
-T
AX=0000 BX=0009 CX=0033 DX=0044 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=010C NU UP EI NG NZ AC PO CY
072A:010C BB3400 MOV BX,0034
```

```

DOSBox Status window
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
-T
AX=0000 BX=0009 CX=0033 DX=0044 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=010C NU UP EI NG NZ AC PO CY
072A:010C BB3400      MOV     BX,0034
-T
AX=0000 BX=0034 CX=0033 DX=0044 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=010F NU UP EI NG NZ AC PO CY
072A:010F B92323      MOV     CX,2323
-T
AX=0000 BX=0034 CX=2323 DX=0044 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0112 NU UP EI NG NZ AC PO CY
072A:0112 F7E3        MUL     BX
-T
AX=0000 BX=0034 CX=2323 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0114 NU UP EI NG ZR AC PO NC
072A:0114 F8          CLC
-T
AX=0000 BX=0034 CX=2323 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0115 NU UP EI NG ZR AC PO NC
072A:0115 F7E1        MUL     CX
-T
AX=0000 BX=0034 CX=2323 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0117 NU UP EI NG ZR AC PO NC
072A:0117 F5          CMC

```

```

DOSBox Status window
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
-T
AX=0000 BX=0034 CX=2323 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0115 NU UP EI NG ZR AC PO NC
072A:0115 F7E1        MUL     CX
-T
AX=0000 BX=0034 CX=2323 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0117 NU UP EI NG ZR AC PO NC
072A:0117 F5          CMC
-T
AX=0000 BX=0034 CX=2323 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0118 NU UP EI NG ZR AC PO CY
072A:0118 F7E2        MUL     DX
-T
AX=0000 BX=0034 CX=2323 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=011A NU UP EI NG ZR AC PO NC
072A:011A B8989F      MOV     AX,9F98
-T
AX=9F98 BX=0034 CX=2323 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=011D NU UP EI NG ZR AC PO NC
072A:011D F7F2        DIV     DX
-T
AX=9F98 BX=0034 CX=2323 DX=0000 SP=FFF8 BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=0192 IP=1DB6 NU UP DI NG ZR AC PO NC
0192:1DB6 2EC706CA548133 MOV     WORD PTR CS:[54CA],3381
CS:54CA=3390

```

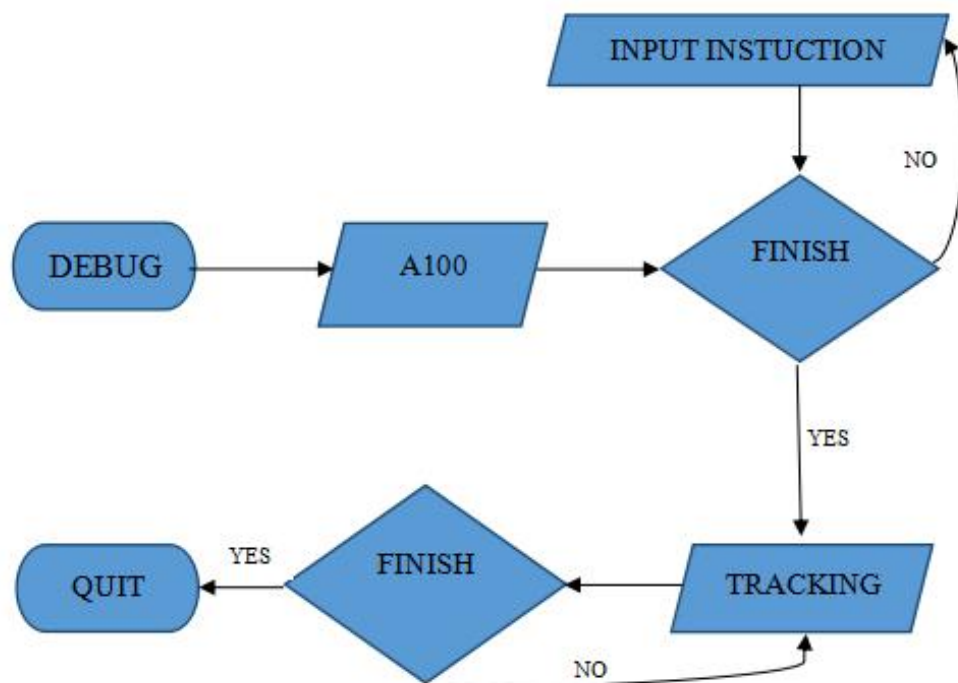
Tabel Data

Alamat	Code	Mnemonic (Operand)	AX		BX		CX		DX		Flag
			AH	AL	BH	BL	CH	CL	DH	DL	
0100		MOV AL,12	00	12	00	00	00	00	00	00	NV UP EI NG NZ NA PO NC
0102	B309	MOV BL,09	00	12	00	09	00	00	00	00	NV UP EI NG NZ NA PO NC
0104	B133	MOV CL,33	00	12	00	09	00	33	00	00	NV UP EI NG NZ NA PO NC
0106	B244	MOV DL,44	00	12	00	09	00	33	00	44	NV UP EI NG NZ NA PO NC
0108	F6E7	MUL BH	00	00	00	09	00	33	00	44	NV UP EI NG ZR NA PO NC
010A	39D8	CMP AX,BX	00	00	00	09	00	33	00	44	NV UP EI NG NZAC PO CY
010C	BB3400	MOV BX,0034	00	00	00	34	00	33	00	44	NV UP EI NG NZ AC PO CY
010F	B92323	MOV CX,2323	00	00	00	34	23	23	00	44	NV UP EI NG NZ AC PO CY
0112	F7E3	MUL BX	00	00	00	34	23	23	00	00	NV UP EI NG ZR AC PO NC
0114	F8	CLC	00	00	00	34	23	23	00	00	NV UP EI NG ZR AC PO NC
0115	F7E1	MUL CX	00	00	00	34	23	23	00	00	NV UP EI NG ZR AC PO NC
0117	F5	CMC	00	00	00	34	23	23	00	00	NV UP EI NG NZ AC PO CY
0118	F7E2	MUL DX	00	00	00	34	23	23	00	00	NV UP EI NGZR AC PO NC
011A	B8989F	MOV AX,9F98	9F	98	00	34	23	23	00	00	NV UP EI NG ZR AC PO NC
011D	F7F2	DIV DX									
011F											
0122											
0125											

G. ALGORITMA PROGRAM

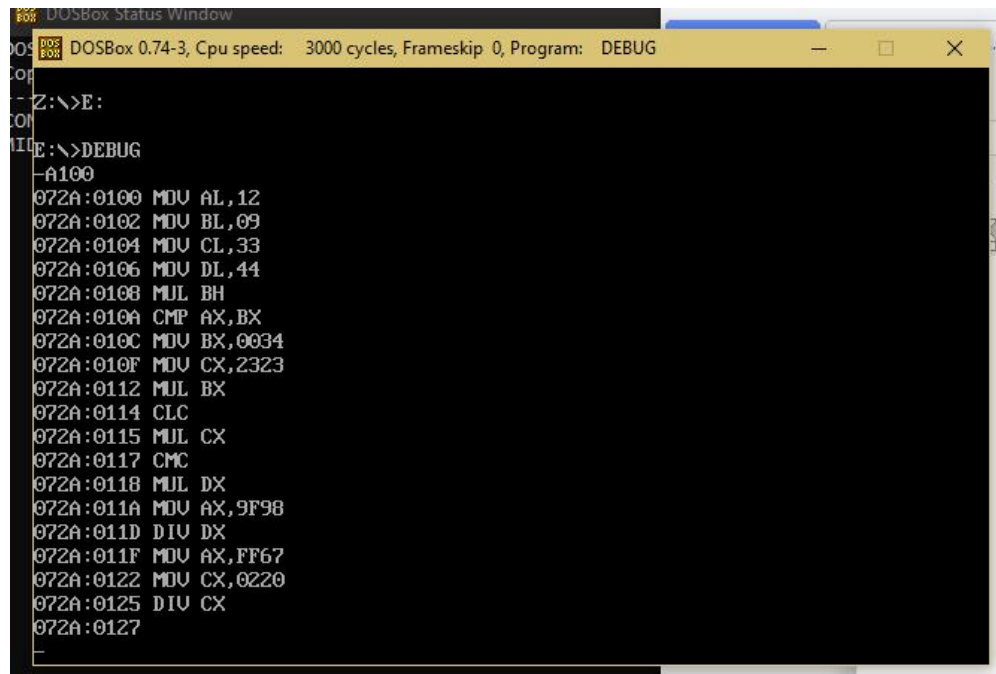
1. Lakukan DEBUG untuk masuk ke sebuah program yang akan dibuat.
2. Masukkan inputan A100 untuk memulai program di alamat offset 0100.
3. Periksa apakah input A100 telah dimasukkan. Jika sudah lanjutkan ke proses Tracking, apabila tidak diinput, lakukan instruksi inputan kembali.
4. Lakukan Tracking.
5. Periksa apakah input sebelumnya telah ditracking. Jika tidak, silahkan ulangi tracking tersebut. Apabila telah selesai tracking, silahkan “QUIT” untuk keluar dari program.

H. FLOWCHART PROGRAM



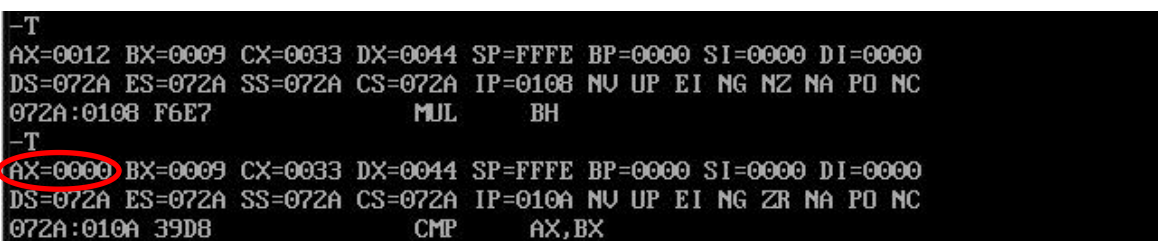
I. ANALISIS PRAKTIKUM

Pada praktikum ini yang melakukan operasi aritmatika dengan menggunakan perintah Operasi Perkalian (MUL) dan Operasi Pembagian (DIV). Perintah-perintah tersebut dapat dilihat pada gambar berikut.



```
DOSBox Status Window
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
Z:\>E:
E:\>DEBUG
-A100
072A:0100 MOV AL,12
072A:0102 MOV BL,09
072A:0104 MOV CL,33
072A:0106 MOV DL,44
072A:0108 MUL BH
072A:010A CMP AX,BX
072A:010C MOV BX,0034
072A:010F MOV CX,2323
072A:0112 MUL BX
072A:0114 CLC
072A:0115 MUL CX
072A:0117 CMC
072A:0118 MUL DX
072A:011A MOV AX,9F9B
072A:011D DIV DX
072A:011F MOV AX,FF67
072A:0122 MOV CX,0220
072A:0125 DIV CX
072A:0127
```

Program dibawah terlihat perintah MUL BH yaitu mengambil nilai pada register BH dan nilai pada AL untuk dikalikan. Dari program dibawah ini sebelumnya nilai AX = 0012, setelah nilai BH = 00 dikalikan dengan AL = 00 maka hasilnya adalah 00 yang akan disimpan pada register AX sehingga AX = 0000.



```
-T
AX=0012 BX=0009 CX=0033 DX=0044 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0108 NU UP EI NG NZ NA PO NC
072A:0108 F6E7 MUL BH
-T
AX=0000 BX=0009 CX=0033 DX=0044 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=010A NU UP EI NG ZR NA PO NC
072A:010A 39D8 CMP AX,BX
```

Selanjutnya pada program dibawah ini adapun perintah CMP (Compare) yang dua buah membandingkan operand. Pada perintah dibawah ini CMP membandingkan AX dan BX tetapi tidak mempengaruhi nilai, perintah CMP hanya akan mempengaruhi flags register sebagai hasil perbandingan. Setelah dilakukan perbandingan antara register AX,BX terjadi perubahan flags. Flag ZR (Zero) menjadi NZ (Not Zero), kemudian NA (Not Auxillary Carry) > AC (Auxillary Carry) dan NC (No Carry) > CY (Carry).

```

-T
AX=0000 BX=0009 CX=0033 DX=0044 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=010A NU UP EI NG ZR NA PO NC
072A:010A 39DB          CMP     AX,BX
-T
AX=0000 BX=0009 CX=0033 DX=0044 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=010C NU UP EI NG NZ AC PO CY
072A:010C BB3400       MOV     BX,0034
-T

```

Berikut dibawah ini adapun perintah CLC (Clear Carry Flag) yang berfungsi untuk membuat Carry Flag menjadi 0. terlihat setelah melakukan perintah CLC terjadi perubahan flag CY (Carry) menjadi NC (No Carry).

```

-T
AX=0000 BX=0034 CX=2323 DX=0044 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0112 NU UP EI NG NZ AC PO CY
072A:0112 F7E3          MUL     BX
-T
AX=0000 BX=0034 CX=2323 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0114 NU UP EI NG ZR AC PO NC
072A:0114 F8           CLC
-T

```

Di bawah ini terdapat perintah CMC (Complement Carry Flag) yaitu melakukan operasi NOT pada Carry Flag. Jika sebelumnya 0 menjadi 1, begitupun sebaliknya.

Terlihat pada gambar setelah melakukan CMC terjadi perubahan flag karena melakukan complement pada flag jika sebelumnya flag NC (Not Carry) > CY (Carry).

```

-T
AX=0000 BX=0034 CX=2323 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0117 NU UP EI NG ZR AC PO NC
072A:0117 F5          CMC
-T
AX=0000 BX=0034 CX=2323 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0118 NU UP EI NG ZR AC PO CY
072A:0118 F7E2          MUL     DX
-T

```

Dan terakhir ada perintah DIV sebagai operasi pembagian. Flags tidak diterangkan setelah DIV.

```

-T
AX=9F98 BX=0034 CX=2323 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=011D NU UP EI NG ZR AC PO NC
072A:011D F7F2          DIV     DX

```

J. KESIMPULAN

Pada praktikum ini, melakukan instruksi dalam proses operasi aritmatika. Perintah-perintah tersebut adalah : MUL yang berfungsi sebagai operasi perkalian, perintah MUL terjadi sesuai dengan jenis perkalian 8 bit atau 16 bit. Dan juga DIV yang merupakan operasi pembagian yang pada dasarnya cara kerjanya sama dengan perkalian (MUL). Sehingga, telah dilakukan praktikum ini didapatkan pemahaman mengenai cara melakukan proses dasar perkalian dan pembagian dalam operasi aritmatika.