

Experiment 7

Name:Pratik Chavan

Div/Batch:A/A1 Roll No.07

.MODEL SMALL

.STACK 100H

.DATA

NUM1 DW 36 ; First number

NUM2 DW 24 ; Second number

GCD_RESULT DW ? ; Store GCD result

LCM_RESULT DW ? ; Store LCM result

MSG_GCD DB 'GCD: \$'

MSG_LCM DB ' LCM: \$'

NEWLINE DB 0DH, 0AH, '\$' ; New line for output formatting

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

MOV AX, NUM1

MOV BX, NUM2

CALL GCD ; Compute GCD

MOV GCD_RESULT, AX

MOV AX, NUM1

MUL BX ; AX = NUM1 * NUM2

DIV GCD_RESULT ; AX = LCM (Product / GCD)

MOV LCM_RESULT, AX

MOV DX, OFFSET MSG_GCD

MOV AH, 09H

INT 21H ; Print "GCD: "

MOV AX, GCD_RESULT

CALL PRINT_NUM ; Print GCD

MOV DX, OFFSET NEWLINE

MOV AH, 09H

INT 21H ; Print new line

MOV DX, OFFSET MSG_LCM

MOV AH, 09H

INT 21H ; Print " LCM: "

MOV AX, LCM_RESULT

CALL PRINT_NUM ; Print LCM

MOV AH, 4CH

INT 21H ; Exit program

MAIN ENDP

; GCD Procedure (Euclidean Algorithm)

GCD PROC

CMP BX, 0

JE END_GCD

GCD_LOOP:

MOV DX, 0

DIV BX ; AX = AX / BX, Remainder in DX

MOV AX, BX

MOV BX, DX

CMP BX, 0

JNE GCD_LOOP

END_GCD:

RET

GCD ENDP

; Print Number Procedure

PRINT_NUM PROC

MOV CX, 0

NEXT_DIGIT:

MOV DX, 0

MOV BX, 10

DIV BX ; AX / 10 → Quotient in AX, Remainder in DX

PUSH DX

INC CX

TEST AX, AX

JNZ NEXT_DIGIT

PRINT_LOOP:

POP DX

ADD DL, '0'

MOV AH, 02H

INT 21H

LOOP PRINT_LOOP

RET

PRINT_NUM ENDP

END MAIN

OUTPUT:

```

CPU 80486 ds:0004 = 0000 3=[↑][↓]=
cs:0003 8ED8      ♦ MOV DS, AX      ax 000C  c=0
cs:0005 A10000    ♦ MOV AX, NUM1    bx 0000  z=1
cs:0008 8B1E0200  ♦ MOV BX, NUM2    cx 0000  s=0
cs:000C E83400    ♦ CALL GCD ; Compute GCD dx 0000  o=0
cs:000F A30400    ♦ MOV GCD_RESULT, AX si 0000  p=1
cs:0012 A10000    ♦ MOV AX, NUM1    di 0000  a=0
cs:0015 F7E3      ♦ MUL BX ; AX = NUM1 * N    bp 0000  i=1
cs:0017 F7360400  ♦ DIV GCD_RESULT ; AX =    sp 0100  d=0
cs:001B A30600    ♦ MOV LCM_RESULT, AX    ds 0884
cs:001E BA0800    ♦ MOV DX, OFFSET MSG_GCD    es 086C
cs:0021 B409      ♦ MOV AH, 09H    ss 0886
cs:0023 CD21      ♦ INT 21H ; Print "GCD:    cs 087C
cs:0025 A10400    ♦ MOV AX, GCD_RESULT    ip 000F

es:0000 CD 20 7D 9D 00 EA FF FF = }¥ Ω
es:0008 AD DE 32 0B C5 05 6B 07 i 2δ+ok•
es:0010 15 03 28 08 15 03 93 01 Š♥(Š♥ô
es:0018 01 01 01 00 02 04 05 06 000 0+•••

ss:0102 0403
ss:0100 52FB
```

Experiment 6

Name:Pratik Chavan

Div/Batch:A/A1 Roll No.07

.MODEL SMALL

.STACK 100H

.DATA

MSG DB 'Flag Register: \$' ; *Message to display*

HEX_CHARS DB '0123456789ABCDEF' ; *Lookup table for hex digits*

FLAGS DW ? ; *Variable to store flag register value*

.CODE

MAIN PROC

MOV AX, DGROUP

MOV DS, AX

PUSHF ; *Push flag register onto the stack*

POP FLAGS ; *Pop flag register into FLAGS variable*

MOV DX, OFFSET MSG

MOV AH, 09H

INT 21H ; *Print "Flag Register: "*

MOV AX, FLAGS ; *Load flag register value into AX*

CALL PRINT_HEX ; *Print the flag register in hexadecimal format*

MOV AH, 4CH

```

    INT 21H    ; Exit program
MAIN ENDP

; -----
; Print 16-bit Hex Procedure
; -----

PRINT_HEX PROC
    MOV CX, 4    ; We have 4 hex digits (16-bit / 4-bit each)
    MOV BX, 12    ; Bit shift amount (12, 8, 4, 0)

HEX_LOOP:
    MOV DX, AX    ; Copy AX value
    MOV CL, BL    ; Move shift count into CL (Fix for SHR error)
    SHR DX, CL    ; Shift right to isolate one hex digit
    AND DX, 0FH    ; Mask the lower 4 bits
    MOV SI, DX    ; Move index to SI
    MOV DL, [HEX_CHARS + SI] ; Convert to ASCII hex character
    MOV AH, 02H
    INT 21H    ; Print the hex digit
    SUB BX, 4    ; Move to the next hex digit
    LOOP HEX_LOOP ; Repeat until all digits are printed

    RET
PRINT_HEX ENDP

END MAIN

```

OUTPUT:

GUI Turbo Assembler x64

File Edit View Run Breakpoints Data Options Window Help

READY

Module: flag File: flag.asm 16

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

3

Experiment 3

Name:Pratik Chavan

Div/Batch:A/A1 Roll No.07

.MODEL SMALL

.STACK 100H

.DATA

NUM DB 5 ; Number for factorial (change this value as needed)

FACT DW 1 ; Variable to store factorial result

MSG DB 'Factorial: \$' ; Message to display before result

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

MOV AL, NUM ; Load number in AL

CBW ; Convert AL to AX (sign-extend)

CALL FACTORIAL ; Call factorial procedure

MOV DX, OFFSET MSG

MOV AH, 09H

INT 21H ; Print message

CALL PRINT_NUM ; Print the factorial result

MOV AH, 4CH


```

    INT 21H      ; Exit program
MAIN ENDP

; Factorial Procedure
FACTORIAL PROC

    MOV CX, AX   ; Move number to CX for loop counter
    MOV AX, 1    ; Initialize AX = 1 (Factorial starts at 1)

FACTORIAL_LOOP:
    MUL CX       ; AX = AX * CX
    LOOP FACTORIAL_LOOP

    MOV FACT, AX ; Store the result in FACT
    RET

FACTORIAL ENDP

; Print Number Procedure
PRINT_NUM PROC

    MOV AX, FACT ; Load factorial result
    MOV CX, 0    ; Clear CX (digit counter)

NEXT_DIGIT:
    MOV DX, 0
    MOV BX, 10

    DIV BX       ; AX / 10 → Quotient in AX, Remainder in DX
    PUSH DX      ; Push remainder (digit) onto stack
    INC CX       ; Increment digit counter
    TEST AX, AX  ; Check if AX is zero

```

JNZ NEXT_DIGIT ; If not, continue extracting digits

PRINT_LOOP:

POP DX ; Get digit from stack

ADD DL, '0' ; Convert to ASCII

MOV AH, 02H

INT 21H ; Print digit

LOOP PRINT_LOOP ; Repeat for remaining digits

RET

PRINT_NUM ENDP

END MAIN

OUTPUT:

The screenshot shows a DOS debugger window with the following content:

CPU 80486		3=1711	
#fact#main		ax 0078	c=0
cs:0000 B88108	♦ MOV AX, DGROUP	bx 0000	z=0
cs:0003 8ED8	♦ MOV DS, AX	cx 0000	s=0
cs:0005 A00000	♦ MOV AL, NUM ; Load num	dx 0000	o=0
cs:0008 98	♦ CBW ; Convert AL to AX	si 0000	p=1
cs:0009 E80E00	♦ CALL FACTORIAL ; Call	di 0000	a=0
cs:000C BA0300	♦ MOV DX, OFFSET MSG	bp 0000	i=1
cs:000F B409	♦ MOV AH, 09H	sp 0100	d=0
cs:0011 CD21	♦ INT 21H ; Print message	ds 0881	
cs:0013 E81100	♦ CALL PRINT_NUM ; Print	es 086C	
cs:0016 B44C	♦ MOV AH, 4CH	ss 0882	
cs:0018 CD21	♦ INT 21H ; Exit program	cs 087C	
#fact#factorial		ip 000C	
es:0000 CD 20 7D 9D 00 EA FF FF = }¥ R			
es:0008 AD DE 32 0B C5 05 6B 07 i 2d *k.			
es:0010 15 03 28 08 15 03 93 01 \$*(S*6@			
es:0018 01 01 01 00 02 04 FF FF @@@ @.			
		ss:0102 0403	
		ss:0100 52FB	

Experiment 3

Name:Pratik Chavan

Div/Batch:A/A1 Roll No.07

.MODEL SMALL

.STACK 100H

.DATA

 ARRAY DB 10, 25, 15, 40, 5, 30, 50, 20

 LEN EQU \$ - ARRAY

 MIN DB 0

 MAX DB 0

.CODE

MAIN PROC

 MOV AX, @DATA

 MOV DS, AX

 MOV SI, 0

 MOV AL, ARRAY[SI]

 MOV MIN, AL

 MOV MAX, AL

FIND_MIN_MAX:

 MOV AL, ARRAY[SI]

 CMP AL, MAX

 JG UPDATE_MAX

 CMP AL, MIN

JL UPDATE_MIN

JMP NEXT_ELEMENT

UPDATE_MAX:

MOV MAX, AL

JMP NEXT_ELEMENT

UPDATE_MIN:

MOV MIN, AL

NEXT_ELEMENT:

INC SI

CMP SI, LEN

JL FIND_MIN_MAX

MOV AX, 4C00H

INT 21H

MAIN ENDP

END MAIN

OUTPUT:

The screenshot shows a debugger window with the following content:

Address	Disassembly	Comment	Register/Value
cs:000D	8A840000	+ mov al, [array + si]	ax 0C07
cs:0011	3AC4	+ cmp al, ah ; Compare c	bx 0000
cs:0013	7F15	+ jg update_max ; If AL	cx 0000
cs:0015	3AC0	+ cmp al, al ; Compare c	dx 0000
cs:0017	7C0D	+ jl update_min ; If AL	si 0006
cs:0019	46	+ inc si	di 0000
cs:001A	A00700	+ mov al, [n] ; Load arr	bp 0000
cs:001D	3BF0	+ cmp si, ax ; Compare i	sp 0100
cs:001F	7CEC	+ jl find_min_max ; If s	ds 0B7F
cs:0021	B8004C	+ mov ax, 4C00h	es 0B6C
cs:0024	CD21	+ int 21h	ss 0B80
#max#update_min			cs 0B7C
			ip 001F

Registers (right side):

c=1
z=0
s=1
o=0
p=1
a=1
i=1
d=0

Stack (bottom):

es:0000	CD 20 7D 9D 00 EA FF FF = 1¥ 2
es:0008	AD DE 32 0B C5 05 6B 07 126+ok
es:0010	15 03 28 0B 15 03 93 01 3*(03*00
es:0018	01 01 01 00 02 04 FF FF 888 8

Stack pointers (bottom right):

ss:0102	0403
ss:0100	52FB