

UITs

UNIVERSITY OF INFORMATION
TECHNOLOGY AND SCIENCES

Assignment on

Lab Project

Course Title

Microprocessor and MicroControllers

Course Code

CSE 360

Submitted by

KM Jakaria

Section: 6A

Batch: 53

ID: 0432310005101037

Submitted to

Md. Ismail

Lecturer

Department of CSE

UITs

Date of Submission

21 Nov, 2025

Experiment Name: Make a Simple Calculator with basic operations like Addition, Subtraction, Multiplication and Division.

Process:

Using 8086 assembly language, this project creates a basic command-line calculator that can be used with the EMU8086 emulator.

One of the four fundamental arithmetic operations is carried out by the calculator once the user enters two integers:

- Addition
- Subtraction
- Multiplication
- Division

Algorithm:

1. Start the program.
2. Initialize the Data Segment.
3. Display a message asking for the first number.
4. Call the number-input procedure to store the number.
5. Display a message asking for the second number.
6. Call the number-input procedure to store the number.
7. Show the menu of operations:
8. 1 → Addition
9. 2 → Subtraction
10. 3 → Multiplication
11. 4 → Division
12. Take the user's choice as a character input.
13. Compare the input with the valid options.
14. According to the choice:
15. Perform addition → store the result.
16. Perform subtraction → store the result.
17. Perform multiplication → store the result.
18. Perform division → store the result.

- 19.If the choice is invalid, display “Invalid choice!”
- 20.Display “Result:” followed by the computed value.
- 21.End the program.

```

edit: C:\Users\Fami\Downloads\calculator.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about

001 ; You may customize this and other start-up templates;
002 ; The location of this template is c:\emu8086\inc\0_com_template.txt
003
004 org 100h
005
006 .model small
007 .stack 100h
008
009 .data
010 msg1 db 10,13,'Enter first number: $'
011 msg2 db 10,13,'Enter second number: $'
012 msg3 db 10,13,'Choose operation:',10,13,'1. Addition',10,13,'2. Subtraction',10,13,'3. Multiplication',10,13,'4. Division',10,13,'Choice: $'
013 msg4 db 10,13,'Result: $'
014 msg5 db 10,13,'Invalid choice!$'
015 newline db 10,13,'$'
016
017 num1 dw ?
018 num2 dw ?
019 result dw ?
020
021 .code
022 main proc
023     mov ax, 0data
024     mov ds, ax
025
026     ; First number input
027     mov ah, 9
028     lea dx, msg1
029     int 21h
030
031     call input_number
032     mov num1, bx
033
034     ; Second number input
035     mov ah, 9
036     lea dx, msg2
037     int 21h
038
039     call input_number
040     mov num2, bx
041
042     ; Operation choice
043     mov ah, 9
044     lea dx, msg3
045     int 21h
046
047     mov ah, 1
048     int 21h
049
050     cmp al, '1'
051     je addition
052     cmp al, '2'
053     je subtraction
054     cmp al, '3'
055     je multiplication
056     cmp al, '4'
057     je division
058
059     ; Invalid choice
060     mov ah, 9
061     lea dx, msg5
062     int 21h
063     jmp exit
064
065 addition:
066     mov ax, num1
067     add ax, num2
068     mov result, ax
069     jmp display_result
070
071 subtraction:
072     mov ax, num1
073     sub ax, num2
074     mov result, ax
075     jmp display_result
076
077 multiplication:
078     mov ax, num1
079     imul num2
080     mov result, ax
081     jmp display_result
082
083 division:
084     mov dx, 0
085     mov ax, num1
086     idiv num2
087     mov result, ax
088     jmp display_result
089
090 display_result:
091     mov ah, 9
092     lea dx, msg4
093     int 21h
094
095     mov ax, result
096     call print_number
097
098 exit:
099     mov ah, 4ch
100     int 21h
101 main endp
102
103 ; Input number procedure
104 input_number proc
105     mov bx, 0
106 input_loop:
107     mov ah, 1
108     int 21h
109     cmp al, 13
110     je input_done
111     sub al, '0'
112     mov cl, al
113     mov ch, 0
114     mov ax, bx
115     mov dx, 10
116     mul dx
117     add ax, cx
118     mov bx, ax
119     jmp input_loop
120 input_done:
121     ret
122 input_number endp
123
124 ; Print number procedure
125 print_number proc
126     mov cx, 0
127     mov bx, 10
128
129     cmp ax, 0
130     jge positive
131     neg ax
132     push ax
133     mov ah, 2
134     mov dl, '-'
135     int 21h
136     pop ax
137
138 positive:
139     mov dx, 0
140     div bx
141     push dx
142     inc cx
143     cmp ax, 0
144     jne positive
145
146 print_loop:
147     pop dx
148     add dl, '0'
149     mov ah, 2
150     int 21h
151     loop print_loop
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

```

Implementation:

Source Codes:

; You may customize this and other start-up templates;

; The location of this template is
c:\emu8086\inc\0_com_template.txt

org 100h

.model small

.stack 100h

.data

msg1 db 10,13,'Enter first number: \$'

msg2 db 10,13,'Enter second number: \$'

msg3 db 10,13,'Choose operation:',10,13,'1. Addition',10,13,'2. Subtraction',10,13,'3. Multiplication',10,13,'4. Division',10,13,'Choice: \$'

msg4 db 10,13,'Result: \$'

msg5 db 10,13,'Invalid choice!\$'

newline db 10,13,'\$'

num1 dw ?

num2 dw ?

result dw ?

.code

main proc

mov ax, @data

mov ds, ax

; First number input

mov ah, 9

lea dx, msg1

int 21h

call input_number

mov num1, bx

; Second number input

mov ah, 9

lea dx, msg2

int 21h

call input_number

mov num2, bx

; Operation choice

mov ah, 9

lea dx, msg3

int 21h

mov ah, 1

int 21h

cmp al, '1'

je addition

cmp al, '2'

je subtraction

cmp al, '3'

je multiplication

cmp al, '4'

je division

; Invalid choice

mov ah, 9

lea dx, msg5

int 21h

jmp exit

addition:

mov ax, num1

add ax, num2

mov result, ax

jmp display_result

subtraction:

mov ax, num1

sub ax, num2

mov result, ax

jmp display_result

multiplication:

mov ax, num1

imul num2	cmp al, 13	mov dx, 0
mov result, ax	je input_done	div bx
jmp display_result	sub al, '0'	push dx
	mov cl, al	inc cx
division:	mov ch, 0	cmp ax, 0
mov dx, 0	mov ax, bx	jne positive
mov ax, num1	mov dx, 10	
idiv num2	mul dx	print_loop:
mov result, ax	add ax, cx	pop dx
jmp display_result	mov bx, ax	add dl, '0'
	jmp input_loop	mov ah, 2
display_result:	input_done:	int 21h
mov ah, 9	ret	loop print_loop
lea dx, msg4	input_number endp	ret
int 21h		print_number endp
	; Print number procedure	
mov ax, result	print_number proc	end main
call print_number	mov cx, 0	
	mov bx, 10	ret
exit:		
mov ah, 4ch	cmp ax, 0	
int 21h	jge positive	
main endp	neg ax	
	push ax	
; Input number procedure	mov ah, 2	
input_number proc	mov dl, '-'	
mov bx, 0	int 21h	
input_loop:	pop ax	
mov ah, 1		
int 21h	positive:	

Method:

1. Taking Input

The program asks the user to enter two numbers.

A custom procedure `input_number` reads multi-digit numbers by:

- Taking input one character at time
- Converting from ASCII to integer
- Building the final number using multiplication by 10

The result is stored in BX and then saved in `num1` and `num2`.

2. Operation Selection

The user selects one of the operations by pressing:

1 Addition

2 Subtraction

3 Multiplication

4 Division

The program reads a single character using `int 21h` (AH = 1) and matches it with the options using `cmp` and `je`.

3. Performing the Operation

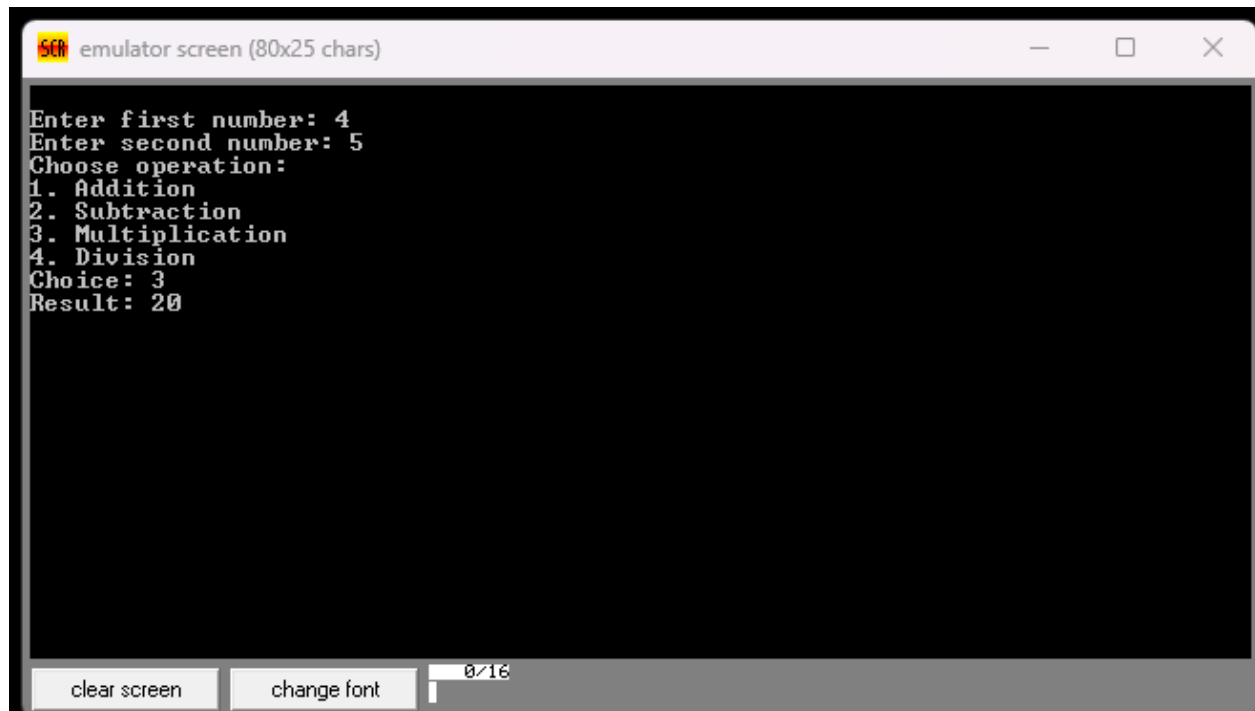
Depending on the choice:

- **Addition:** `ADD AX, num2`
- **Subtraction:** `SUB AX, num2`
- **Multiplication:** `IMUL num2`
- **Division:**
 - Clear DX
 - `IDIV num2`

The result is stored in the variable `result`.

Result:

The application displays as like user friendly calculator



Conclusion:

This project effectively illustrates how simple arithmetic operations can be carried out using assembly language.

This calculator application teaches us:

- Using interruptions to handle user input
- Putting multi-digit numeric input into practice
- Using CPU registers to do arithmetic
- Using bespoke processes to display numbers
- Conditional branching in menu-based systems

The project offers a solid basis for comprehending low-level programming ideas and the direct hardware execution of tasks.