Proyecto Assembler

Gutierrez Morales Erick Hugo

INF153A-Assembler-Informática-FCPN-UMSA

09 de Junio de 2024

```
1
2  ; Materia: INF153 Assembler
3  ; Estudiante: Erick Gutierrez Morales
4
5  mescolx macro texto, color
6      mov al, texto
7      mov bl, color
8      mov cx, 35
9      mov ah, 9
10     int 10h
11     mov ah, 9
12     mov dx, offset texto
13     int 21h
14 endm
15
16 messaje macro x
17     lea dx, x
18     mov ah, 9
19     int 21h
20 endm
21
22 data segment
23     v db 40 dup(0)
24     enter db 10,13,36
25     ms1 db "ES PRIMO$"
26     ms2 db "NO ES PRIMO$"
27     ms3 db "NUMERO FELIZ$"
28     ms4 db "NUMERO INFELIZ$"
29     ms5 db "NUMERO PERFECTO$"
30     ms6 db "NUMERO IMPERFECTO$"
31     ms7 db  "+---------------------------------+$"
32     ms8 db  "|  cmd  |       descripcion       |$"
33     ms9 db  "+-------+-------------------------+$"
34     ms10 db "|   b   |  fibonacci              |$"
35     ms11 db "|   f   |  factorial              |$"
36     ms12 db "|   v   |  maximo comun divisor   |$"
37     ms13 db "|   m   |  minimo comun multiplo  |$"
38     ms14 db "|   d   |  divisores              |$"
39     ms15 db "|   p   |  primos menores a       |$"
40     ms16 db "|   e   |  es primo               |$"
41     ms17 db "|   c   |  cuadrado               |$"
42     ms18 db "|   u   |  cubo                   |$"
43     ms19 db "|   t   |  potencia               |$"
44     ms20 db "|   r   |  raiz cuadrada          |$"
```

```
    ms21 db "|    z    |  raiz                     |$"
    ms22 db "|    x    |  promedio                 |$"
    ms23 db "|    l    |  numero feliz             |$"
    ms24 db "|    n    |  numero perfecto          |$"
    ms25 db "|    *    |  muestra el menu          |$"
    ms26 db "+--------+---------------------------+$"
    aux dw 0
    num dw 0
    ans dw 0
    sum dw 0
    cnt db 0
    cnu db 0
    cnv db 0
    a dw 0
    b dw 0
    x dw 0
    i dw 0
    n db 0
ends

stack segment
    dw   128  dup(0)
ends

code segment
start:
    ; set segment registers:
    mov ax, data
    mov ds, ax
    ;mov es, ax

    ; add your code here
    mov ax, 0
    mov cx, 0
    mov cl, es:[128]
    cmp cx, 0
    jz fin
    ; posicion con el metodo que se desea
    mov si, 130
    mov al, es:[si]
    add si,  2
    sub cl, 2
    cmp al, 98
    jz fibonacci
    cmp al, 102
    jz factorial
    cmp al, 118
    jz maximoComunDivisor
    cmp al, 109
    jz minimoComunMultiplo
```

```
95      cmp al, 100
96      jz divisores
97      cmp al, 112
98      jz primosMenoresA
99      cmp al, 101
100     jz esPrimo
101     cmp al, 99
102     jz cuadrado
103     cmp al, 117
104     jz cubo
105     cmp al, 116
106     jz potencia
107     cmp al, 114
108     jz raizCuadrada
109     cmp al, 122
110     jz raiz
111     cmp al, 120
112     jz promedio
113     cmp al, 108
114     jz numeroFeliz
115     cmp al, 110
116     jz numeroPerfecto
117     cmp al, 42
118     jz menu
119     jnz fin
120     ;************************************
121     menu:
122     mescolx ms7, 11
123     messaje enter
124     mescolx ms8, 11
125     messaje enter
126     mescolx ms9, 11
127     messaje enter
128     mescolx ms10, 11
129     messaje enter
130     mescolx ms11, 11
131     messaje enter
132     mescolx ms12, 11
133     messaje enter
134     mescolx ms13, 11
135     messaje enter
136     mescolx ms14, 11
137     messaje enter
138     mescolx ms15, 11
139     messaje enter
140     mescolx ms16, 11
141     messaje enter
142     mescolx ms17, 11
143     messaje enter
144     mescolx ms18, 11
```

```
145     messaje enter
146     mescolx ms19, 11
147     messaje enter
148     mescolx ms20, 11
149     messaje enter
150     mescolx ms21, 11
151     messaje enter
152     mescolx ms22, 11
153     messaje enter
154     mescolx ms23, 11
155     messaje enter
156     mescolx ms24, 11
157     messaje enter
158     mescolx ms25, 11
159     messaje enter
160     mescolx ms26, 11
161     jmp fin
162
163     ;***********************************
164     fibonacci:
165     call readOneInput
166     mov cx, bx
167     mov a, -1
168     mov b, 1
169     fibo:
170         mov bx, a
171         add bx, b
172         mov ax, b
173         mov a, ax
174         mov b, bx
175         push cx
176         mov ans, bx
177         call print
178         pop cx
179     loop fibo
180     jmp fin
181
182     ;***********************************
183     factorial:
184     call readOneInput
185     mov cx, 0
186     mov cl, bl
187     mov bx, 1
188     mov x, 1
189     cmp cl, 0
190     jz printFactorial
191     facto:
192         mov ax, bx
193         mov bx, x
194         mul bx
```

```
195         mov bx, ax
196         inc x
197     loop facto
198     printFactorial:
199     mov ans, bx
200     call print
201     jmp fin
202 
203     ;***********************************
204     maximoComunDivisor:
205     call readTwoInput
206     mcd:
207         mov dx, 0
208         mov ax, a
209         mov bx, b
210         div bx
211         mov ax, b
212         mov b, dx
213         mov a, ax
214         cmp b, 0
215         jz endMcd
216     jnz mcd
217     endMcd:
218     mov bx, a
219     mov ans, bx
220     call print
221     jmp fin
222 
223     ;***********************************
224     minimoComunMultiplo:
225     call readTwoInput
226     mov x, 0
227     mov ax, a
228     mov bx, b
229     mul bx
230     mov x, ax
231     mcm:
232         mov dx, 0
233         mov ax, a
234         mov bx, b
235         div bx
236         mov ax, b
237         mov b, dx
238         mov a, ax
239         cmp b, 0
240         jz endMcm
241     jnz mcm
242     endMcm:
243     mov ax, x
244     mov bx, a
```

```
245     div bx
246     mov ans, ax
247     call print
248     jmp fin
249 
250     ;*********************************
251     divisores:
252     call readOneInput
253     mov aux, bx
254     mov num, bx
255     inc num
256     mov cx, 1
257     divider:
258         mov dx, 0
259         mov ax, aux
260         mov bx, cx
261         div bx
262         cmp dx, 0
263         jz printDivider
264         jnz searchDivider
265         printDivider:
266             mov ans, cx
267             call print
268         searchDivider:
269         inc cx
270         cmp cx, num
271         jz endDivider
272     jnz divider
273     endDivider:
274     jmp fin
275 
276     ;*********************************
277     primosMenoresA:
278     call readOneInput
279     mov cx, bx
280     mov aux, 1
281     for:
282         mov bx, aux
283         mov num, bx
284         call checkPrime
285         cmp x, 0
286         jz noEsPrimo
287         mov ax, aux
288         mov ans, ax
289         call print
290         noEsPrimo:
291         inc aux
292     loop for
293     jmp fin
294 
```

```
295     ;***********************************
296     esPrimo:
297     call readOneInput
298     mov num, bx
299     call checkPrime
300     cmp x, 0
301     jz printFalse
302     jnz printTrue
303     printFalse:
304     messaje ms2
305     jmp fin
306     printTrue:
307     messaje ms1
308     jmp fin
309
310     ;***********************************
311     cuadrado:
312     call readOneInput
313     mov ax, bx
314     mul bx
315     mov ans, ax
316     call print
317     jmp fin
318
319     ;***********************************
320     cubo:
321     call readOneInput
322     mov ax, bx
323     mul bx
324     mul bx
325     mov ans, ax
326     call print
327     jmp fin
328
329     ;***********************************
330     potencia:
331     call readTwoInput
332     mov bx, 1
333     power:
334         mov ax, bx
335         mov bx, a
336         mul bx
337         mov bx, ax
338         dec b
339         cmp b, 0
340         jz endPower
341     jnz power
342     endPower:
343     mov ans, bx
344     call print
```

```
345     jmp fin
346
347     ;***********************************
348     raizCuadrada:
349     call readOneInput
350     mov cx, 0
351     cmp bx, 0
352     jz save
353     cmp bx, 1
354     jz save
355     mov i, 1
356     mov ax, 1
357     cmp ax, bx
358     ja save
359     root:
360         inc i
361         mov ax, i
362         mov cx, i
363         mul cx
364         cmp ax, bx
365         ja endRoot
366     jbe root
367     endRoot:
368     dec i
369     mov ax, i
370     mov ans, ax
371     jmp checkDecimal
372     save:
373     mov ans, bx
374
375     checkDecimal:
376     ; obtenemos el primer residuo
377     mov cx, bx
378     mov ax, i
379     mov bx, i
380     mul bx
381     sub cx, ax
382     ; aniadimos 2 ceros para los decimales
383     mov ax, cx
384     mov bx, 100
385     mul bx
386     mov cx, ax
387     ; iteramos para encontrar el 1er decimal
388     mov ax, i
389     mov bx, 20
390     mul bx
391     call getDecimal
392     mov ax, i
393     mov a, ax
394     ; obtenemos el segundo residuo
```

```
395     mov ax, cx
396     sub ax, x
397     ; aniadimos dos ceros para los decimales
398     mov bx, 100
399     mul bx
400     mov cx, ax
401     ; iteremos para encontrar el 2do decimal
402     mov ax, ans
403     mov bx, 10
404     mul bx
405     add ax, a
406     mov bx, 20
407     mul bx
408     call getDecimal
409     mov ax, i
410     mov b, ax
411     ; imprimimos la parte entera
412     mov cnv, 1
413     call print
414     ; imprimimos el punto decimal
415     mov dl, 46
416     mov ah, 2
417     int 21h
418     ; imprimimos el primer decimal
419     mov ax, a
420     mov ans, ax
421     call print
422     ; imprimimos el segundo decimal
423     mov ax, b
424     mov ans, ax
425     call print
426     jmp fin
427 
428     ;*********************************
429     raiz:
430     call readTwoInput
431     cmp a, 0
432     jz imprimir
433     cmp a, 1
434     jz imprimir
435     mov i, 1
436     mov cx, 0
437     radical:
438         mov ax, 1
439         mov cx, b
440         ciclo:
441             mov bx, i
442             mul bx
443         loop ciclo
444         cmp ax, a
```

```
445         ja endRadical
446         inc i
447     jbe radical
448     endRadical:
449     dec i
450     imprimir:
451     mov ax, i
452     mov ans, ax
453     call print
454     jmp fin
455 
456     ;*********************************
457     promedio:
458     mov cnt, cl
459     mov ax, 0
460     mov bx, 0
461     mov cx, 0
462     mov num, 0
463     mov sum, 0
464     mov cnu, 0
465     forPromedio:
466         mov cl, es:[si]
467         cmp cl, 13
468         jz endPromedio
469         cmp cl, 32
470         jz set
471         jnz componer
472         set:
473         add sum, bx
474         inc cnu
475         mov bx, 0
476         jmp noComponer
477         componer:
478         sub cl, 48
479         mov ax, bx
480         mov bx, 10
481         mul bx
482         add ax, cx
483         mov bx, ax
484         noComponer:
485         inc si
486         dec cnt
487         cmp cnt, 0
488         jz endPromedio
489     jnz forPromedio
490     endPromedio:
491     add sum, bx
492     inc cnu
493 
494     mov ax, 0
```

```
    mov bx, 0
    mov dx, 0
    ; parte entera
    mov ax, sum
    mov bl, cnu
    div bx
    mov cnv, 1
    mov ans, ax
    push dx
    call print
    ; impresion del punto decimal
    mov dl, 46
    mov ah, 2
    int 21h
    pop dx
    ; obtenemos los decimales
    mov cnt, 2
    forDec:
        mov ax, dx
        mov bx, 10
        mul bx
        mov dx, 0
        mov bl, cnu
        div bx
        push dx
        mov dl, al
        add dl, 48
        mov ah, 2
        int 21h
        pop dx
        dec cnt
        cmp cnt, 0
        jz endForDec
    jnz forDec
    endForDec:
    jmp fin

    ;*********************************
    numeroFeliz:
    call readOneInput
    mov num, bx
    ; verificamos si es un numero feliz
    mov si, offset v
    mov cx, 0
    mov n, 0
    happyNumber:
        mov sum, 0
        mov cnt, 0
        while1:
            mov dx, 0
```

```
545             mov ax, bx
546             mov bx, 10
547             div bx
548             push dx
549             inc cnt
550             cmp ax, 0
551             jz endWhile1
552             mov bx, ax
553         jnz while1
554         endWhile1:
555         while2:
556             pop ax
557             mov bx, ax
558             mul bx
559             add sum, ax
560             dec cnt
561             cmp cnt, 0
562             jz endWhile2
563         jnz while2
564         endWhile2:
565         cmp sum, 1
566         jz endHappyNumber
567         ; verificamos que no exista sum en el
568         ; array y guardamos.
569         mov di, offset v
570         mov ax, sum
571         mov cl, n
572         cmp cx, 0
573         jz pass
574         forArray:
575             cmp al, [di]
576             jz endHappyNumber
577             inc di
578         loop forArray
579         pass:
580         mov ax, sum
581         mov [si], ax
582         inc si
583         inc n
584         ; actualizamos el valor de bx
585         mov bx, sum
586     jnz happyNumber
587     endHappyNumber:
588     cmp sum, 1
589     jz verdad
590     jnz falso
591     verdad:
592         messaje ms3
593         jmp fin
594     falso:
```

```
595         messaje ms4
596         jmp fin
597 
598     ;*********************************
599     numeroPerfecto:
600     call readOneInput
601     mov dx, 0
602     mov cx, 0
603     mov num, bx
604     mov sum, 0
605     mov cx, 1
606     findDivider:
607         mov dx, 0
608         mov ax, num
609         mov bx, cx
610         div bx
611         cmp dx, 0
612         jz igual
613         jnz noigual
614         igual:
615         add sum, cx
616         noigual:
617         inc cx
618         cmp cx, num
619         jz endFindDivider
620     jnz findDivider
621     endFindDivider:
622     mov ax, num
623     mov bx, sum
624     cmp ax, bx
625     jz perfect
626     jnz notPerfect
627     perfect:
628         messaje ms5
629         jmp fin
630     notPerfect:
631         messaje ms6
632 
633     ;*********************************
634     fin:
635     ; exit to operating system
636     mov ax, 4c00h
637     int 21h
638 
639     ;*********************************
640     ; subrutina para leer un solo numero
641     readOneInput:
642     mov ax, 0
643     mov bx, 0
644     mov cnu, cl
```

```
645     mov cx, 0
646     oneInput:
647         mov cl, es:[si]
648         cmp cl, 13
649         jz endOneInput
650         sub cl, 48
651         mov ax, bx
652         mov bx, 10
653         mul bx
654         add ax, cx
655         mov bx, ax
656         dec cnu
657         cmp cnu, 0
658         jz endOneInput
659         inc si
660     jnz oneInput
661     endOneInput:
662     ret
663 
664     ; subrutina para leer dos numeros
665     readTwoInput:
666     mov ax, 0
667     mov bx, 0
668     mov cnu, cl
669     mov cx, 0
670     twoInput:
671         mov cl, es:[si]
672         cmp cl, 13
673         jz endTwoInput
674         cmp cl, 32
675         jz reset
676         jnz compose
677         reset:
678         mov a, bx
679         mov ax, 0
680         mov bx, 0
681         mov cx, 0
682         jmp decompose
683         compose:
684         sub cl, 48
685         mov ax, bx
686         mov bx, 10
687         mul bx
688         add ax, cx
689         mov bx, ax
690         decompose:
691         dec cnu
692         cmp cnu, 0
693         jz endTwoInput
694         inc si
```

```
695     jnz twoInput
696     endTwoInput:
697     mov b, bx
698     ret
699 
700     ; subrutina verifica si es Primo
701     checkPrime:
702     cmp num, 1
703     jbe false
704     cmp num, 2
705     jz true
706     mov dx, 0
707     mov ax, num
708     mov bx, 2
709     div bx
710     cmp dx, 0
711     jz false
712     mov i, 3
713     mov ax, i
714     mov bx, i
715     mul bx
716     cmp ax, num
717     ja true
718     prime:
719         mov dx, 0
720         mov ax, num
721         mov bx, i
722         div bx
723         cmp dx, 0
724         jz false
725         add i, 2
726         mov ax, i
727         mov bx, i
728         mul bx
729         cmp ax, num
730         ja true
731     jbe prime
732     false:
733         mov x, 0
734         jmp continue
735     true:
736         mov x, 1
737     continue:
738     ret
739 
740     ; subrutina obtener decimal
741     getDecimal:
742     mov i, 1
743     decimal:
744         push ax
```

```
745         add ax, i
746         mov bx, i
747         mul bx
748         cmp ax, cx
749         jae endDecimal
750         mov x, ax
751         pop ax
752         inc i
753     jb decimal
754     endDecimal:
755     dec i
756     pop ax
757     ret
758 
759     ; subrutina mostrar
760     print:
761     mov cnt, 0
762     getDigit:
763         mov dx, 0
764         mov ax, ans
765         mov bx, 10
766         div bx
767         push dx
768         inc cnt
769         mov ans, ax
770         cmp ans, 0
771         jz endGetDigit
772     jnz getDigit
773     endGetDigit:
774     mostrar:
775         pop dx
776         add dl, 48
777         mov ah, 2
778         int 21h
779         dec cnt
780         cmp cnt, 0
781         jz finMostrar
782     jnz mostrar
783     finMostrar:
784     cmp cnv, 1
785     jz notSpace
786     mov dl, 32
787     mov ah, 2
788     int 21h
789     notSpace:
790     ret
791 ends
792 
793 end start
```