תוכניות דוגמא gcd4.asm, fib1.asm, fib2.asm, fib3.asm, fib3.asm

התוכנית gcd4.asm הינה מימוש בשיטה ה"יעילה" של הלולאת ה-while בתוכנית gcd3.asm, שהיא המימוש ה"אינטואיטיבי" שלה. כלומר ב-gcd4.asm הבדיקה בסוף הלולאה ומסתעפים אליה בהתחלה.

התוכניות fib1.asm, fib2.asm, fib3.asm, fib4.asm ממחישות מימוש for לחישוב מספרי פיבונצי. fib1.asm, fib2.asm עושות זאת בדרך "הכללית" fib3.asm ו-fib4.asm נעזרת בפקודה LOOP ו-LOOPD.

fib1.asm מממש את הלולאה בדרך ה"אינטואיטבית" ו-fib2.asm בדרך ה"יעילה". בהערות יש את האלגוריתם בקוד C שקוד האסמבלי הוא כביכול "התרגום" שלה.

fib4.asm משתמש בפקודות הלולאה המורחבות של ה-386. שימו לב שהלולאה נשלטת על ידי ECX ולא CX, האגר ECX הוא שמקבל את n ושהפקודות JECXZ ו-LOOPD החליפו את JCX ו-LOOP.

```
;
; gcd4.asm - Compute greatest common divisor, 386 version
;
     .MODEL SMALL
     .STACK 100h
     .DATA
X    DD  881790
Y    DD  188955
Gcd  DD  ?
     .CODE
     .386             ;  Enable 386 code
     MOV AX,@DATA     ; Program prefix
     MOV DS,AX        ; Set DS to point to data segment
                      ;
     MOV EAX,X         ;  Fisrt operand
     MOV EBX,Y         ;  Second operand
     JMP TestNext      ;                                  ***** C *****
Do1:                   ;                                while (eax != ebx)
                       ;                                  {
                       ;                                       if (eax<ebx)
       JAE Xhigh       ; Skip XCHG IF EAX=>EBX                   {
         XCHG EAX,EBX  ; EAX < EBX, Swap them                     temp = eax;
                       ;                                          eax = ebx;
                       ;                                          ebx = temp;
       Xhigh:          ;                                         }
         SUB EAX,EBX   ; EAX := EAX - EBX                      eax = eax - ebx;
TestNext:              ;
     CMP EAX,EBX       ;
     JNE Do1           ; Exit if EAX = EBX
Endlp:                 ;                                   } /* while */
     MOV Gcd,EAX       ; Store result                     gcd = eax;
                      ;
     MOV AH,4Ch       ; Set terminate option for int 21h
     INT 21h          ; Return to DOS (terminate program)
     END
```

```
;
; fib1.asm - Compute Fibonacci n
;
     .MODEL SMALL
     .STACK 100h
     .DATA
n        DD  20
Fibo_n   DD  ?
      .CODE
      .386              ;  Enable 386 code
      MOV AX,@DATA      ; Program prefix
      MOV DS,AX         ; Set DS to point to data segment
                        ;
      MOV ECX,n          ;              *****  C  *****
      MOV ESI,3          ;              ecx = n;
      MOV EBX,1          ;
      MOV EDX,1          ;              ebx = edx = 1;
Do1:                     ;              for(esi = 3; esi <= ecx; esi++)
      CMP ESI,ECX        ;               {
      JA Endlp           ;                 eax = ebx + edx;
      MOV EAX,EBX        ;                 edx = ebx;
      ADD EAX,EDX        ;                 ebx = eax;
                         ;               }
      MOV EDX,EBX        ;
      MOV EBX,EAX         ;
                          ;
      INC ESI             ;
      JMP Do1
                          ;
Endlp:                    ;
      MOV Fibo_n,EAX      ; Store result    fibo_n = eax
                        ;
      MOV AH,4Ch        ; Set terminate option for int 21h
      INT 21h           ; Return to DOS (terminate program)
      END
```

```
;
; fib2.asm - Compute Fibonacci n
;
     .MODEL SMALL
     .STACK 100h
     .DATA
n        DD  20
Fibo_n   DD  ?
      .CODE
      .386              ;  Enable 386 code
      MOV AX,@DATA      ; Program prefix
      MOV DS,AX         ; Set DS to point to data segment
                        ;
      MOV ECX,n          ;              *****  C  *****
      MOV ESI,3          ;              ecx = n;
      MOV EBX,1          ;
      MOV EDX,1          ;              ebx = edx = 1;
      JMP TestNext       ;
Do1:                     ;              for(esi = 3; esi <= ecx; esi++)
      MOV EAX,EBX        ;                {
      ADD EAX,EDX        ;                  eax = ebx + edx;
                         ;                  edx = ebx;
      MOV EDX,EBX        ;                  ebx = eax;
      MOV EBX,EAX         ;               }
                          ;
      INC ESI             ;
                          ;
TestNext:                 ;
      CMP ESI,ECX         ;
      JNA Do1             ;
Endlp:                    ;
      MOV Fibo_n,EAX      ; Store result    fibo_n = eax
                        ;
      MOV AH,4Ch        ; Set terminate option for int 21h
      INT 21h           ; Return to DOS (terminate program)
      END
```

```
;
; fib3.asm - Compute  Fibonacci n
;
     .MODEL SMALL
     .STACK 100h
     .DATA
n        DW  20
Fibo_n   DD  ?
      .CODE
      .386             ;  Enable 386 code
      MOV AX,@DATA     ; Program prefix
      MOV DS,AX        ; Set DS to point to data segment
                       ;
      MOV CX,n         ;              *****  C  *****
      SUB CX,2         ;               cx = n-2;
      JS  Endlp        ;
      MOV EBX,1         ;
      MOV EDX,1         ;               eax = ebx = edx = 1;
      MOV EAX,1         ;
      JCXZ EndLp        ;
Do1:                    ;               for(cx = n-3; cx > 0; cx-- )
      MOV EAX,EBX       ;                 {
      ADD EAX,EDX       ;                    eax = ebx + edx;
                        ;                    edx = ebx;
      MOV EDX,EBX       ;                    ebx = eax;
      MOV EBX,EAX        ;                 }
                         ;
      LOOP Do1           ;
                         ;
Endlp:                   ;
      MOV Fibo_n,EAX     ; Store result      fibo_n = eax
                        ;
      MOV AH,4Ch        ; Set terminate option for int 21h
      INT 21h           ; Return to DOS (terminate program)
      END
```

```
;
; fib4.asm - Compute fibo
;
     .MODEL SMALL
     .STACK 100h
     .DATA
n        DD  20
Fibo_n   DD  ?
      .CODE
      .386              ;  Enable 386 code
      MOV AX,@DATA      ; Program prefix
      MOV DS,AX         ; Set DS to point to data segment
                        ;
      MOV ECX,n          ;              *****  C  *****
      SUB ECX,2          ;
      JS Endlp           ;
      MOV EBX,1          ;
      MOV EDX,1          ;               eax = ebx = edx = 1;
      MOV EAX,1          ;
      JECXZ EndLp        ;
Do1:                     ;               for(ecx = n-2; ecx > 0; ecx-- )
      MOV EAX,EBX        ;                 {
      ADD EAX,EDX        ;                    eax = ebx + edx;
                         ;                    edx = ebx;
      MOV EDX,EBX        ;                    ebx = eax;
      MOV EBX,EAX         ;                 }
                          ;
      LOOPD Do1           ;
                          ;
Endlp:                    ;
      MOV Fibo_n,EAX      ; Store result      fibo_n = eax
                        ;
      MOV AH,4Ch        ; Set terminate option for int 21h
      INT 21h           ; Return to DOS (terminate program)
      END
```

JXX - Jump Instructions Table

Mnemonic	Meaning	Jump Condition
JA	Jump if Above	CF=0 and ZF=0
JAE	Jump if Above or Equal	CF=0
JB	Jump if Below	CF=1
JBE	Jump if Below or Equal	CF=1 or ZF=1
JC	Jump if Carry	CF=1
JCXZ	Jump if CX Zero	CX=0
JE	Jump if Equal	ZF=1
JG	Jump if Greater (signed)	ZF=0 and SF=OF
JGE	Jump if Greater or Equal (signed)	SF=OF
JL	Jump if Less (signed)	SF != OF
JLE	Jump if Less or Equal (signed)	ZF=1 or SF != OF
JMP	Unconditional Jump	unconditional
JNA	Jump if Not Above	CF=1 or ZF=1
JNAE	Jump if Not Above or Equal	CF=1
JNB	Jump if Not Below	CF=0
JNBE	Jump if Not Below or Equal	CF=0 and ZF=0
JNC	Jump if Not Carry	CF=0
JNE	Jump if Not Equal	ZF=0
JNG	Jump if Not Greater (signed)	ZF=1 or SF != OF
JNGE	Jump if Not Greater or Equal (signed)	SF != OF
JNL	Jump if Not Less (signed)	SF=OF
JNLE	Jump if Not Less or Equal (signed)	ZF=0 and SF=OF
JNO	Jump if Not Overflow (signed)	OF=0
JNP	Jump if No Parity	PF=0
JNS	Jump if Not Signed (signed)	SF=0
JNZ	Jump if Not Zero	ZF=0
JO	Jump if Overflow (signed)	OF=1
JP	Jump if Parity	PF=1
JPE	Jump if Parity Even	PF=1
JPO	Jump if Parity Odd	PF=0
JS	Jump if Signed (signed)	SF=1
JZ	Jump if Zero	ZF=1