Example of the ‘LGrind’ Package

John Leis

15 June 1996

Language	Command in DOS batch file
C	`lgrind -i -v subst %1.c > %1.lg`
MASM	`lgrind -i -lmasm %1.asm > %1.lg`

Figure 1: Commands for lgrind'ing source file into $\LaTeX\,2_\varepsilon$ format

```c
/* endian.c
 * Demonstrates endian ordering
 */

#include <stdio.h>

void main( void )
{
        short  Data_16;
        long   Data_32;
        char far *p;

        Data_16 = 0x1234;
        Data_32 = 0x56789abc;

        p = (char far *)&Data_16;
        printf("16-bit quantity, data=%04x\n", Data_16);
        printf("address %Fp = %02x\n", p, (int)(*p) & 0xff);
        p++ ;
        printf("address %Fp = %02x\n", p, (int)(*p) & 0xff);
        p++ ;


        p = (char far *)&Data_32;
        printf("32-bit quantity, data=%08lx\n", Data_32);
        printf("address %Fp = %02x\n", p, (int)(*p) & 0xff);
        p++ ;
        printf("address %Fp = %02x\n", p, (int)(*p) & 0xff);
        p++ ;
        printf("address %Fp = %02x\n", p, (int)(*p) & 0xff);
        p++ ;
        printf("address %Fp = %02x\n", p, (int)(*p) & 0xff);

}
```

Figure 2: Example 'C' language program.

This is a multi-page listing

It has no caption.
Used for Appendices etc.

```
;***********************************************
; vgac.asm
; PC VGA graphics control in assembly language
; uses BIOS for keyboard read and setting graphics
; modes, and procedure for setting a VGA pixel
; version for C calling convention :-
;       LARGE model
;       no MAIN entry point
;       assemble only (no link)
;       underscore for C-callable functions
;       don't pop arguments off stack (caller does this)
;
; J Leis
; 24 May 1994
;***********************************************

TITLE  vgac.asm - vga assembler program, callable from C
.MODEL LARGE
.286
.DOSSEG

; stack segment directive
.STACK

; data segment directive
.DATA

; code segment directive
.CODE


_VgaMode        PROC

   pusha
   mov ah, 0  ; function 0 = set video mode
   mov al, 12h ; mode 12 = vga graphics
   int 10h
   popa
   ret

_VgaMode   ENDP


_TextMode PROC

   pusha
   mov ah, 0  ; function 0 = set video
   mov al, 03h ; mode 3 = text
   int 10h
   popa
   ret

_TextMode       ENDP


_ShowMessage   PROC
```

```
   pusha     ; save registers if necessary

   ; call DOS interrupt to display a message
   mov bx, 01h
   lea dx, mesg      ; equivalent to mov dx, OFFSET mesg
   mov cx, l_mesg
   mov ah, 040h
   int 021h

   popa
   ret

_ShowMessage   ENDP


_ReadKey       PROC

   pusha     ; save registers if necessary

   mov ah, 00h ; function 0 - wait for key & read it
   int 16h   ; int 16h = keyboard services
   ; al now equals ascii code of key

   popa
   ret

_ReadKey       ENDP


; setpixel( xc, yc, color )
; stacking order:
;           memory  near call far call
;  color    highest [bp+8]    [bp+10]
;  y-coord          [bp+6]    [bp+8]
;  x-coord  lowest  [bp+4]    [bp+6]
;
_SetPixel      PROC

   push bp
   mov bp, sp

   pusha     ; save registers if necessary

   mov dx, 03CEh          ; graphics controller register

   mov ax, 0205h          ; write mode 2
   out dx, ax

   mov ax, 0003h          ; function
   out dx, ax

   mov ax, 0A000h         ; graphics screen segment
   mov es, ax

   mov ax, [bp+8]        ; get y co-ord
   mov bx, 640/8          ; 80 bytes/line
   mul bx
   mov bx, [bp+6]        ; get x-coord
   mov cl, 3              ; divide by 8 bits/byte
   shr bx, cl
   add bx, ax
   mov al, es:[bx]        ; dummy write to latch data in screen RAM
   mov cx, [bp+6]        ; get x-coord
   and cx, 0007h          ; get bit mask
   mov al, 07h
   sub al, cl
   mov ah, 80h
   shr ah, cl              ; shift to bit position
   mov al, 08h             ; set mask register

   mov dx, 03CEh           ; dx destroyed by mul
   out dx, ax              ; write bit mask

   mov cx, [bp+10]; color  ; write the color value
   mov es:[bx], cl

   popa

   pop bp
   ret          ; don't pop args off stack – C does this

_SetPixel          ENDP

;no  main procedure (main in C)

; end of file
END
```