3.

```
        global  _main
        extern  _glClear@4
        extern  _glBegin@4
        extern  _glEnd@0
        extern  _glColor3f@12
        extern  _glVertex3f@12
        extern  _glFlush@0
        extern  _glutInit@8
        extern  _glutInitDisplayMode@4
        extern  _glutInitWindowPosition@8
        extern  _glutInitWindowSize@8
        extern  _glutCreateWindow@4
        extern  _glutDisplayFunc@4
        extern  _glutMainLoop@0

        section .text
title:  db      'A Simple Triangle', 0
zero:   dd      0.0
one:    dd      1.0
half:   dd      0.5
neghalf:dd      -0.5

display:
        push    dword 16384
        call    _glClear@4              ; glClear(GL_COLOR_BUFFER_BIT)
```

```
        push    dword 9
        call    _glVertex3f@12          ; glVertex(-.5, -.5, 0)
        push    dword 0
        push    dword [one]
        push    dword 0
```

4.

```
call    _glColor3f@12           ; glColor3f(0, 0, 1)
        push    dword 0
        push    dword [half]
        push    dword 0
        call    _glVertex3f@12          ; glVertex(0, .5, 0)
        call    _glEnd@0                ; glEnd()
        call    _glFlush@0              ; glFlush()
        ret
_main:
        push    dword [esp+8]           ; push argv
        lea     eax, [esp+8]            ; get addr of argc (offset changed
:-)
        push    eax
        call    _glutInit@8             ; glutInit(&argc, argv)
        push    dword 0
        call    _glutInitDisplayMode@4
        push    dword 80
        push    dword 80
        call    _glutInitWindowPosition@8
        push    dword 300
```

```
push    dword 400
call    _glutInitWindowSize@8
push    title
call    _glutCreateWindow@4
push    display
call    _glutDisplayFunc@4
call    _glutMainLoop@0
ret
```

5. a. Addressing modes are an aspect of the instruction set architecture in most central processing unit (CPU) designs. The various addressing modes that are defined in a given instruction set architecture define how machine language instructions in that architecture identify the operand(s) of each instruction. An addressing mode specifies how to calculate the effective memory address of an operand by using information held in registers and/or constants contained within a machine instruction or elsewhere.

b. Indirect Addressing: indirect addressing is a very powerful addressing mode which in many cases provides an exceptional level of flexibility. Indirect addressing is also the only way to access the extra 128 bytes of Internal RAM found on an 8052. Since indirect addressing always refers to Internal RAM these two instructions would write the value 01h to Internal RAM address 99h on an 8052. On an 8051 these two instructions would produce an undefined result since the 8051 only has 128 bytes of Internal RAM

c. Immediate Addressing: Immediate addressing is so-named because the value to be stored in memory immediately follows the operation code in memory. That is to say, the instruction itself dictates what value will be stored in memory.However, since the value to be loaded is fixed at compile-time it is not very flexible.

d. Direct Addressing - Direct addressing is so-named because the value to be stored in memory is obtained by directly retrieving it from another memory location. Direct addressing is generally fast since, although the value to be loaded isn't included in the instruction, it is quickly accessible since it is stored in the 8051's Internal RAM. It is also much more flexible than Immediate Addressing since the value to be loaded is whatever is found at the given address--which may be variable. Also, it is important to note that when using direct addressing any instruction which refers to an address between 00h and 7Fh is referring to Internal Memory. Any instruction which refers to an address between 80h and FFh is referring to the SFR control registers that control the 8051 microcontroller itself.