CS F363 COMPILER CONSTRUCTION

LAB-01

1. Familiarize with gcc options.

$ man gcc (-E, -S, -O<n>, -g<n>, -m32 )

1. Use the C grammar to explain the order of evaluation of the expression.

|  |  |
| --- | --- |
| **a)**  main()  {  int a, b, c;  float x, y;  x = y += (a >> 4) + b % c;  } |  |

1. What is the output of the program. guess before executing it.

|  |  |
| --- | --- |
| /\* file f1.c \*/  #include <stdio.h>  #define SQR(x) (x \* x)  main()  {  int a = 5, b = 3, c;  c = SQR(a);  printf("c=%d\n", c);  c = SQR(a+b);  printf("c=%d\n", c);  } | 1. Are the outputs correct. 2. Use gcc –E f1.c > f1.i (vi f1.i and find out) 3. How will you correct it? ( look at file f1.i ) |

1. Study –S, –O optimization flags, -m32 flag, -fno-asynchronous-unwind-tables

|  |  |
| --- | --- |
| $ gcc -S 3.c  int a = 4, b = 5;  main()  {  int c = a + b;  printf("%d\n", c);  } | -O0(default), -O1/2/3   (optimization options)  $gcc –m32 -O2 a.c -o a-O2.out  $gcc –m32 -O2 -S a.c; cp a.s a-O2.s  $gcc –m32 -O0 -S a.c  compare a.s and a-O2.s |

1. Same prog as in 3. Study –g<n> flag (debug) , CTRL-D to exit from debugger

|  |
| --- |
| $ gcc –m32 4.c -o 4.out  $ gcc –m32 –g3 4.c –o 4-dbg.out  $ ls –l (compare the size of executables)  $ gdb 4.out  (gbd) list (what is the o/p)  $ gdb 4-dbg.out  (gdb) list  (gdb) break main  (gdb) run  (gdb) step , stepi, next, nexti  (gdb) print c  (gdb) disassemble |

1. Analyze and explain the execution time of program. (time command)

|  |  |  |
| --- | --- | --- |
| /\* 5.c \*/  #include <stdio.h>  int A[10000][10000];  main()  {    int i, j, n = 10000;    for (i=0; i<n; i++)      for (j=0; j<n; j++)        A[i][j] = 0;  } | $gcc 5.c -o 5.out  $time 5.out  $ cp 1.c 2.c  $ vi 2.c  Replace A[i][j] = 0;  with A[j][i] = 0;  $gcc 2.c -o 2.out  $time 2.out | $ cp 1.c 3.c  $ vi 3.c  int A[10000][10000]  int \*ap = A;    replace A[i][j] = 0  with \*ap++ = 0;  $ gcc 3.c -o 3.out  $ time 3.out |

1. Analyze and explain the output of the program

|  |  |
| --- | --- |
| int a[5] = { 1, 2, 3, 4, 5};  int i = 2, j;  main()  {  j = a[++i] + a[++i];  printf("%d\n", j);    } | $ gcc a.c -o a  $ ./a  $ gcc -O2 a.cc -o a-O2  $./a-O2   1. Explain why there is a difference (if any). 2. Is it acceptable? Is the compiler doing any mistake.? |

1. Study the assembly output of the program (-S, -O2)

|  |  |
| --- | --- |
| int a[10],[b[10],c[10];  int i;  main()  {  for(i=0;i<10;i++)  a[i] = b[i] + c[i];  }  $ gcc –m32 –S a.c  $ gcc –m32 –S –O2 a.c | Modify the program as follows.  int a[10],[b[10],c[10];  int \*ap = a, \*bp = b, \*cp = c;  main()  {  for(i=0;i<10;i++)  \*ap++ = \*bp++ + \*cp++;  }  $ gcc –m32 –S a.c  $ gcc –m32 –S –O2 a.c |

1. Study the assembly output of the program (-S, -O2)

|  |  |
| --- | --- |
| int a, b, c, d;  main()  {  a = 0; b = 1; c = 0;  d = a && b || c; printf("%d", d);  a = 1; b = 0; c = 1;  d = a && b || c; printf("%d", d);  } | $ gcc –m32 a.c –o a.out  ./a.out (explain the output)  $ gcc –m32 –S a.c |

1. Functions, parameter passing, structs, unions (use –S, , gdb )

|  |  |
| --- | --- |
| int a=10,b=20,c = 30, d;  int func(int x, int y, int \*z)  {  int a;  a = x + y \* \*z;  return a;  }  main()  {    d = func(a, b, &c);    } | **b)**  typedef  struct point { int x, int y; } POINT  POINT addPoints(POINT a, POINT \*b)  {  POINT c;  c.x = a.x + b->x;  c.y = a.y + b->y;  a.x++;  b->x++;  return c;  }  main()  {  POINT p1 = {2, 3}, p2 = {4, 5}, p3;  p3 = addPoints(p1, &p2);  //printf  } |
| **c)**  Consider an integer no (32 bits). Extract the bytes (0,1,2,3) of the int variable;  (at least) 3 possible ways are there ☺ | |

1. Replace the switch code with function pointers (enjoy the brevity of code)

|  |  |
| --- | --- |
| int a=10,b=20,c;  int op; //0..3  int add(int, int);  int sub(int, int);  int mul(int, int);  int mul(int, int);  main()  {    switch( op) {     case 0 : c = add(a,b); break;     case 1 : c = sub(a,b); break;     case 2 : c = mul(a,b); break;     case 3 : c = div(a,b); break;  }  printf("%d", c);    }  int add(int a, int b) { return a + b; }  int sub(int a, int b) { return a - b; }  int mul(int a, int b) { return a \* b; }  int div(int a, int b) { return a / b; } | Replaced code will look like this  main()  {  // line 1:  //variable declaration of func pointer  //line 2:  //invoke the function add,sub,mul,div  //based on op  } |

1. Analyse the code using gdv

|  |  |
| --- | --- |
| main()  {  int \*p = 100;  \*p= 20;  printf("d\n", \*p);  }  $gcc –m32 a.c  $./a.out  Segmentation fault (core dumped)  $ls –l  $ulimit –a  $ulimit –c ulimited  $gcc –m32 –g3 a.c  $a.out  $ls –l | $ gcc -m32 -g3 gb.c -o gb  gb.c: In function ‘main’:  gb.c:4:13: warning: initialization makes pointer from integer without a cast [enabled by default]  int \*p = 100;  ^  gb.c:6:4: warning: incompatible implicit declaration of built-in function ‘printf’ [enabled by default]  printf("d\n", \*p);  ^  $ ./gb  Segmentation fault (core dumped)  $ ls -l  total 156  -rw------- 1 skv users 208896 Jan 25 06:01 core.21904  -rwxr-xr-x 1 skv users 15835 Jan 25 06:01 gb  -rw-r--r-- 1 skv users 70 Jan 25 06:00 gb.c  $gdb gb core.21904  Core was generated by `./gb'.  Program terminated with signal 11, Segmentation fault.  #0 0x08048435 in main () at gb.c:5  5 \*p= 20; |

1. Is this a legal C Code Analyze using C Grammar

|  |
| --- |
| send(to, from, count)  register short \*to, \*from;  register count;  {  register n = count / 8;  do {  \*to = \*from++;  \*to = \*from++;  \*to = \*from++;  \*to = \*from++;  \*to = \*from++;  \*to = \*from++;  \*to = \*from++;  \*to = \*from++;  } while (--n > 0);  }  Ref: <https://en.wikipedia.org/wiki/Duff%27s_device>  http://www.drdobbs.com/a-reusable-duff-device/184406208?queryText=%2522duff%2527s%2Bdevice%2522 |