CS 302 : Implementation of Programming Languages
TUTORIAL 10 (Compilation of OO features); April 13, 2017

The following five programs are used in the problems in this tutorial sheet. The programs and their assembled codes are uploaded on moodle.

Program 1 : A C++ program with inheritance and virtual functions.

```
class Base
{ public:
   int b;
   virtual int f1(int a) {return b+a;};
   virtual int f2(int i, int j) {return i*b+j;};
   virtual int f3() { return b*b;}
};

class D1: public Base { public: int d1;
   int f1(int x) { return d1+x;};
};

class D2: public D1{ public: int d2;
   int f2(int k, int l) {return d1+d2+k+l;};
};

int main()
{ int s1, s2, s3;
  s1 = sizeof(Base); s2 = sizeof(D1); s3 = sizeof(D2);
  cout << s1 << "  " << s2 << "  " << s3 << endl;
  Base b; D1 dd1, dd12; D2 dd2, dd22;
  b.b = 5; dd1.b = 10; dd1.d1 = 11; dd12.b = 15;
  dd12.d1 = 16;  dd2.b = 50; dd2.d1 = 51; dd2.d2 = 52;
  dd22.b = 60; dd22.d1 = 61; dd22.d2 = 62;
  s1 = b.f1(5); s2 = b.f2(10, 15);
  cout << s1 << "  " << s2 << endl;
  s1 = dd1.f1(20); s2 = dd1.f2(30, 35);
  cout << s1 << "  " << s2 << endl;
  s1 = dd2.f1(40); s2 = dd2.f2(50, 100);
  cout << s1 << "  " << s2 << endl;
  int res, k=0;
  Base * aa[5];
  aa[0] = &dd1; aa[1] = &dd2; aa[2] = &b;
  aa[3] = &dd12; aa[4] = &dd22;
  for ( k; k < 5; k++)
  { res = aa[k]->f1(k);
   cout << " aa[ " << k << "]->f1( " << k << ") = "
   << res << endl; }
  D1 * bb[4];
  bb[0] = &dd1; bb[1] = &dd12; bb[2] = &dd2;
  bb[3] = &dd22;
  for ( k = 0; k < 4; k++)
  { res = bb[k]->f1(k);
    cout << " bb[ " << k << "]->f1( "
         << k << ") = " << res << endl;};
}
```

Program 2 : Same as Program 1 except for the fact that all functions defined in all the classes are qualified a s virtual. The code that differs is displayed below.

```
class D1: public Base { public: int d1;
   virtual int f1(int x) { return d1+x;};
};

class D2: public D1{ public: int d2;
   virtual int f2(int k, int l) {return d1+d2+k+l;};
};
```

Program 3: Same as Program 1 except for the fact that the function f1() defined in class D1 is the only virtual function in the program. The code that differs is displayed below.

```
class Base
{ public:  int b;
   int f1(int a) {return b+a;};
   int f2(int i, int j) {return i*b+j;};
   int f3() { return b*b;}
};

class D1: public Base { public: int d1;
   virtual int f1(int x) { return d1+x;};
};
class D2: public D1{ public: int d2;
   int f2(int k, int l) {return d1+d2+k+l;};
};
```

Program 4: Class definitions are identical to that of Program 1. The only difference is in the main() function, wherein the last for-loop (which invokes f1() on array elements bb[]) has been changed. The code that differs is displayed below.

In Program 1 :

```
for ( k = 0; k < 4; k++)
  { res = bb[k]->f1(k);
     cout << " bb[ " << k << "]->f1( "
            << k << ") = " << res << endl;};
```

In Program 4:

```
for ( k = 0; k < 4; k++)
{ res = bb[k]->f2(k, k+1);
  cout << " bb[ " << k << "]->f2( " << k << ", "
         << k+1  <<  ") = " << res << endl;  };
```

Program 5:

```
class Base
{ public:
   int b;
   int f1(int a) {return b+a;};
   int f2(int i, int j) {return i*b+j;};
};
class D1: public Base { public: int d1;
   virtual int f1() { return d1+b;};
};
class D2: public D1 { public: int d2;
   virtual int f1(int k) {return d2+k;};
};
int main()
{ int s1, s2, s3;
  s1 = sizeof(Base); s2 = sizeof(D1); s3 = sizeof(D2);
  cout << s1 << "  " << s2 << "  " << s3 << endl;
  Base b; D1 dd1, dd12; D2 dd2, dd22;
  b.b = 5; dd1.b = 10; dd1.d1 = 11; dd12.b = 15; dd12.d1
= 16;
  dd2.b = 50; dd2.d2 = 52; dd22.b = 60; dd22.d2 = 62;
  s1 = b.f1(5);
  s2 = dd1.f2(20, 25);
  s3 = dd2.f2(50, 100);
  Base * aa[3];
  aa[0] = &dd1; aa[1] = &dd2; aa[2] = &b;
  int res, k=0;
  for (k; k < 3; k++)
  { res = aa[k]->f1(k);
    cout << " aa[ " << k << "]->f1( " << k << ") = "
         << res << endl;
  }
}
```

P1.

(a) Manually execute all the five programs and write their outputs with reasons.
(b) Examine the compilation model discussed in the class for each of the programs and comment on whether it is adequate or it needs to be changed with reasons.
(c) Show the contents of the virtual function tables of each class for each of the programs.

The answer to part (a) is given at the end of the tutorial sheet. Check your manually computed results with this and refresh these concepts for C++.

P2. Examine part of the assembly code of function main() of Program 1 given in the following.

(a) Justify the presence of "$224" in the assembly instruction : **subq $224, %rsp**

```
main:
.LFB976:
      pushq   %rbp    #
      movq    %rsp, %rbp      #,
      subq    $224, %rsp      #,
      movl    $16, -208(%rbp) #, s1
      movl    $16, -204(%rbp) #, s2
      movl    $24, -200(%rbp) #, s3
      movl    -208(%rbp), %eax        # s1, tmp128
      movl    %eax, %esi      # tmp128,
      movl    $_ZSt4cout, %edi        #,
      …............
```

(b) For each call in the assembly code, identify the corresponding source code equivalent.

(c) Show the layout of all variables/objects of main() in its activation record, the contents of the stack from %rbp to %rsp in terms of the source program entities.

(d) Annotate the assembly code of main() with the source program entities with respect to compilation

(e) Identify the assembly code for the following two statements in main() which are in the two for-loops and show the contents of the activation record just before and after both these function calls.

```
res = aa[k]->f1(k);
…........
res = bb[k]->f1(k);
```

P3. (a) Examine the Assembly code of all functions of Program 1, except for the main(). Explain why so many functions have been created and what is the role of each function.

(b) Explain the purpose of the assembly code, **movq -8(%rbp), %rax** that is used in several of these functions.

(c) Identify the assembly code and the corresponding source code for all function calls that are resolved statically by the compiler. Provide reasons as to why the compiler used compile time resolution for these calls..

(d) Identify the assembly code and the corresponding source code that correspond to use of the virtual function tables (constructed by the compiler) in the resolution of a function call and explain your answer.

(e) Several functions have an assembly instruction of the form : **subq $16, %rsp** in their body while there are other functions that don't. Explain the reason for this difference. Also explain the reason for the constant, such as $16 in the instruction above, wherever it occurs.

P4. Consider Program 2 given above. Examine the assembly code of main() of this program and compare it with the corresponding assembly code of Program 1. Report the similarity and differences.

(b) Identify the assembly code and the corresponding source code for all function calls that are resolved statically by the compiler. Provide reasons as to why the compiler used compile time resolution for these calls..

(c) Identify the assembly code and the corresponding source code that correspond to use of the virtual function tables (constructed by the compiler) in the resolution of a function call and explain your answer.

P5. Examine the assembly code of function main() of Program 3.
(a) Identify the assembly code and the corresponding source code for all function calls that are resolved statically by the compiler. Provide reasons as to why the compiler used compile time resolution for these calls..
(b) Identify the assembly code and the corresponding source code that correspond to use of the virtual function tables (constructed by the compiler) in the resolution of a function call and explain your answer.

P6. Examine the assembly code of function main() of Program 4..
(a) For each call in the assembly code, identify the corresponding source code equivalent.
(b) Show the layout of all variables/objects of main() in its activation record, the contents of the stack from %rbp to %rsp in terms of the source program entities.
(c) Annotate the assembly code of main() with the source program entities with respect to compilation
(d) Identify the assembly code for the following two statements in main() which are in the two for-loops and show the contents of the activation record just before and after both these function calls.
res = aa[k]->f1(k);
…........
res = bb[k]->f2(k, k+1);

P7. Examine the assembly code of function main() of Program 5.
(a) Annotate the assembly code of main() with the source program entities with respect to compilation
(b) Identify the assembly code for the statement, **res = aa[k]->f1(k);** , in main() which are in the for-loop and show the contents of the activation record just before and after both this function call.
(c) Identify the assembly code and the corresponding source code for all function calls that are resolved statically by the compiler. Provide reasons as to why the compiler used compile time resolution for these calls.
(d) Identify the assembly code and the corresponding source code that correspond to use of the virtual function tables (constructed by the compiler) in the resolution of a function call and explain your answer.

******* End of Tutorial Sheet 10 ********

Supratim Biswas

ANSWERS TO COMMON PROBLEM PART (a)

Program 1:
```
16  16  24
10  65
31  335
91  253
 aa[ 0]->f1( 0) = 11
 aa[ 1]->f1( 1) = 52
 aa[ 2]->f1( 2) = 7
 aa[ 3]->f1( 3) = 19
 aa[ 4]->f1( 4) = 65
 bb[ 0]->f1( 0) = 11
 bb[ 1]->f1( 1) = 17
 bb[ 2]->f1( 2) = 53
 bb[ 3]->f1( 3) = 64
```

Program 2:
```
16  16  24
10  65
31  335
91  253
 aa[ 0]->f1( 0) = 11
 aa[ 1]->f1( 1) = 52
 aa[ 2]->f1( 2) = 7
 aa[ 3]->f1( 3) = 19
 aa[ 4]->f1( 4) = 65
 bb[ 0]->f1( 0) = 11
 bb[ 1]->f1( 1) = 17
 bb[ 2]->f1( 2) = 53
 bb[ 3]->f1( 3) = 64
```

Program 3:
```
4  16  24
10  65
31  335
91  253
 aa[ 0]->f1( 0) = 10
 aa[ 1]->f1( 1) = 51
 aa[ 2]->f1( 2) = 7
 aa[ 3]->f1( 3) = 18
 aa[ 4]->f1( 4) = 64
 bb[ 0]->f1( 0) = 11
 bb[ 1]->f1( 1) = 17
 bb[ 2]->f1( 2) = 53
 bb[ 3]->f1( 3) = 64
```

Program 4:
```
16  16  24
10  65
31  335
91  253
 aa[ 0]->f1( 0) = 11
 aa[ 1]->f1( 1) = 52
 aa[ 2]->f1( 2) = 7
 aa[ 3]->f1( 3) = 19
 aa[ 4]->f1( 4) = 65
 bb[ 0]->f2( 0, 1) = 1
 bb[ 1]->f2( 1, 2) = 17
 bb[ 2]->f2( 2, 3) = 108
 bb[ 3]->f2( 3, 4) = 130
```

Program 5:
```
 4  16  24
 aa[ 0]->f1( 0) = 10
 aa[ 1]->f1( 1) = 51
 aa[ 2]->f1( 2) = 7
```