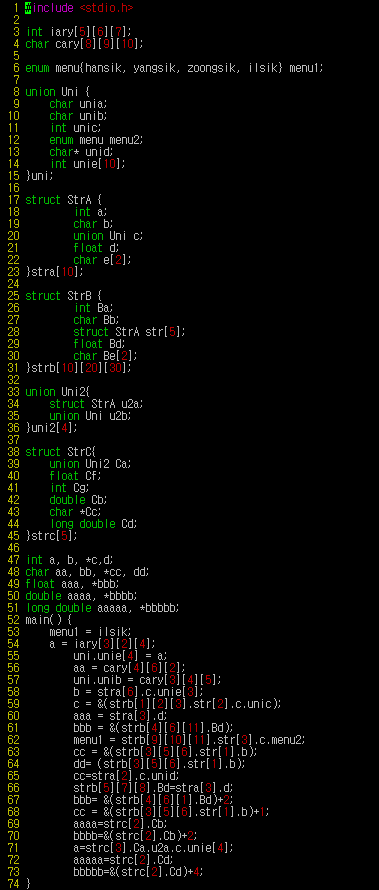
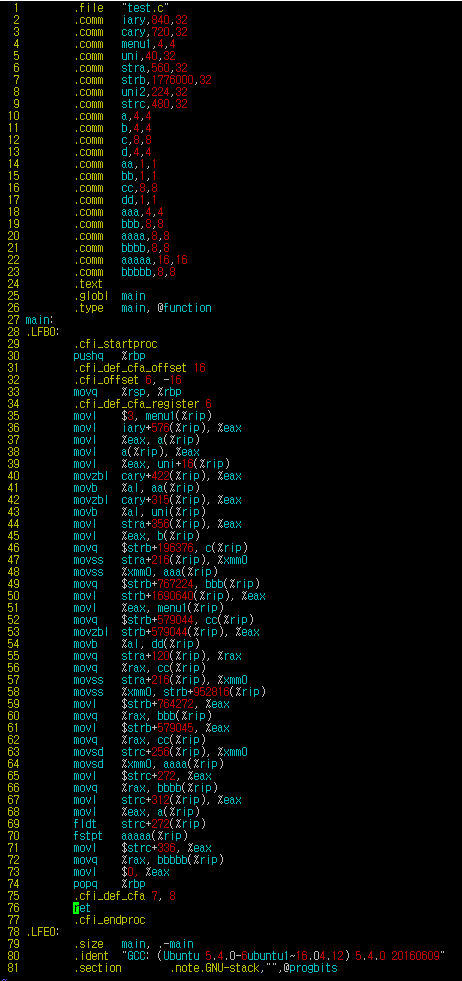
**프로그래밍언어 과제4**

**20161595 배성현**



(전체 코드)



어셈블리 코드

|  |  |
| --- | --- |
| int iary[5][6][7];  char cary[8][9][10];  enum menu{hansik, yangsik, zoongsik, ilsik} menu1;  union Uni {  char unia;  char unib;  int unic;  enum menu menu2;  char\* unid;  int unie[10];  }uni;  struct StrA {  int a;  char b;  union Uni c;  float d;  char e[2];  }stra[10];  struct StrB {  int Ba;  char Bb;  struct StrA str[5];  float Bd;  char Be[2];  }strb[10][20][30];  union Uni2{  struct StrA u2a;  union Uni u2b;  }uni2[4];  struct StrC{  union Uni2 Ca;  float Cf;  int Cg;  double Cb;  char \*Cc;  long double Cd;  }strc[5];  int a, b, \*c,d;  char aa, bb, \*cc, dd;  float aaa, \*bbb;  double aaaa, \*bbbb;  long double aaaaa, \*bbbbb; | .comm iary,840,32  .comm cary,720,32  .comm menu1,4,4  .comm uni,40,32  .comm stra,560,32  .comm strb,1776000,32  .comm uni2,224,32  .comm strc,480,32  .comm a,4,4  .comm b,4,4  .comm c,8,8  .comm d,4,4  .comm aa,1,1  .comm bb,1,1  .comm cc,8,8  .comm dd,1,1  .comm aaa,4,4  .comm bbb,8,8  .comm aaaa,8,8  .comm bbbb,8,8  .comm aaaaa,16,16  .comm bbbbb,8,8 |
| 먼저 왼쪽은 c코드에서 데이터를 전역변수로 정의한 부분이고, 오른쪽은 이 c코드가 어셈블리코드로 바뀐 것에서의 데이터가 정의된 부분이다. | |
| **int iary[5][6][7];** -> **.comm iary,840,32** (5\*6\*7\*4(int형 크기)=840byte) | |
| **char cary[8][9][10];** -> **.comm cary,720,32** (8\*9\*10\*1(char형)=720byte) | |
| **enum menu{hansik, yangsik, zoongsik, ilsik} menu1;** -> **.comm menu1,4,4** (4byte(enumeration형)) | |
| **union Uni {**  **char unia;**  **char unib;**  **int unic;**  **enum menu menu2;**  **char\* unid;**  **int unie[10];**  **}uni;**  -> .**comm uni,40,32**  (union Uni에서 가장 큰 공간을 차지하는 데이터는 int unie[10]이다. union은 가장 큰 공간을 차지하는 데이터와 같은 크기로 할당이 되기 때문에, 4(int)\*10=40byte) | |
| **struct StrA {**  **int a;**  **char b;**  **union Uni c;**  **float d;**  **char e[2];**  **}stra[10];**  **-> .comm stra,560,32**  먼저 struct StrA타입의 크기는 4(int)+1(char)+3(word alignment)+40(Uni)+4(float)+1(char)\*2+2(word alignment)=56byte이다. 따라서 stra는 56(StrA)\*10(개수)=560byte이다. | |
| **struct StrB {**  **int Ba;**  **char Bb;**  **struct StrA str[5];**  **float Bd;**  **char Be[2];**  **}strb[10][20][30];**  **-> .comm strb,1776000,32**  먼저 struct StrB형의 크기는 4(int)+1(char)+3(word alignment)+56(StrA)\*5+4(float) +1(char)\*2+2(word alignment)=296byte이다. 따라서 strb는 296(StrB)\*10\*20\*30(개수)=1776000byte이다.. | |
| **union Uni2{**  **struct StrA u2a;**  **union Uni u2b;**  **}uni2[4];**   * **.comm uni2,224,32**   위에서 struct StrA 타입의 크기를 구했을 때 56byte 였고, union Uni 타입의 크기는 40byte였다. 따라서 StrA타입의 크기가 더 크기 때문에 union Uni2타입의 크기는 56byte가 된다. 이러한 union Uni2타입의 배열 uni2[4]는 56\*4=224byte이다. | |
| **struct StrC{**  **union Uni2 Ca;**  **float Cf;**  **int Cg;**  **double Cb;**  **char \*Cc;**  **long double Cd;**  **}strc[5];**   * **.comm strc,480,32**   위에서 union Uni2타입의 크기를 구했을 때 56byte였다. 따라서 struct StrC타입의 크기는 56(Uni2)+4(float)+4(int)+8(double)+8(char\*)+16(long double)=96byte이다. 이러한 struct StrC타입의 배열 strc[5]는 96\*5=480byte이다. | |
| **int a, b, \*c,d;**  **char aa, bb, \*cc, dd;**  **float aaa, \*bbb;**  **double aaaa, \*bbbb;**  **long double aaaaa, \*bbbbb;**  **->**  **.comm a,4,4**  **.comm b,4,4**  **.comm c,8,8**  **.comm d,4,4**  **.comm aa,1,1**  **.comm bb,1,1**  **.comm cc,8,8**  **.comm dd,1,1**  **.comm aaa,4,4**  **.comm bbb,8,8**  **.comm aaaa,8,8**  **.comm bbbb,8,8**  **.comm aaaaa,16,16**  **.comm bbbbb,8,8**  a, b, d는 int형이므로 4바이트를 할당 받고, aa, bb, dd는 char형이므로 1바이트를 할당 받는다. aaa는 float 형이므로 4바이트를 할당 받고 aaaa는 double형이므로 8바이트를 할당 받는다. 또한 aaaaa는 long double형이므로 16바이트를 할당받는다. 그리고 마지막으로 c, cc, bbb, bbbb, bbbbb는 포인터이므로 8byte를 할당 받는다. | |

**c코드와 대응되는 어셈블리 코드**

|  |
| --- |
| menu1 = ilsik;  movl $3, menu1(%rip)  enumeration type이 integer로 저장이 되는데, ilsik은 enumeration type menu에서 4번째로 선언한 원소이다. 값을 0부터 가지게 되므로, 0부터 4번째는 3이므로 ilsik의 값은 3이고 따라서 menu1에 $3이 movl되는 것을 확인 할 수 있다. |
| a = iary[3][2][4];  movl iary+576(%rip), %eax  movl %eax, a(%rip)  iary[3][2][4]의 주소는 iary의 주소로부터 4(int형)\*(3\*6\*7+2\*7+4)(c언어는 row major order이므로)만큼 떨어져있게 된다. 이를 계산하면 576이 되고, 따라서 iary+576에 있는 값을 eax로 movl하고 eax를 다시 a로 movl함으로써 a = iary[3][2][4];을 만족하게 된다. |
| uni.unie[4] = a;  movl a(%rip), %eax  movl %eax, uni+16(%rip)  uni.unie[4]의 주소는 unie가 int형 배열이므로 uni의 주소로부터 4(int형)\*4만큼 떨어져있게 된다. 따라서 uni+16이 uni.unie[4]의 주소이므로, a에 있는 값을 eax에 movl하고, eax를 다시 uni+16에 movl함으로써 uni.unie[4] = a;를 만족하게 된다. |
| aa = cary[4][6][2];  movzbl cary+422(%rip), %eax  movb %al, aa(%rip)  cary[4][6][2]의 주소는 cary의 주소로부터 1(char형)\*(4\*9\*10+6\*10+2)(c언어는 row major order이므로)만큼 떨어져있게 된다. 이를 계산하면 422가 되고, 따라서 cary+422에 있는 값을 eax로 옮기고 al을 다시 aa로 옮김으로써 aa = cary[4][6][2];을 만족하게 된다. |
| uni.unib = cary[3][4][5];  movzbl cary+315(%rip), %eax  movb %al, uni(%rip)  cary[3][4][5]의 주소는 cary의 주소로부터 1(char형)\*(3\*9\*10+4\*10+5)(c언어는 row major order이므로)만큼 떨어져있게 된다. 이를 계산하면 315가 된다. 또한 uni.unib의 주소는 uni의 주소와 같다. 따라서 cary+315에 있는 값을 eax로 옮기고 al을 다시 uni로 옮김으로써 uni.unib = cary[3][4][5];을 만족하게 된다. |
| b = stra[6].c.unie[3];  movl stra+356(%rip), %eax  movl %eax, b(%rip)  먼저 struct StrA 타입의 크기는 56byte이다. 따라서 stra[6].c.unie[3]의 주소는 stra+56(StrA)\*6(인덱스 전 원소개수)+4(int)+1(char)+3(word alignment)+4(int)\*3(인덱스 전 원소개수)이므로 str+356이다. 따라서 stra+356에 있는 값을 eax로 옮기고 eax를 다시 b로 옮김으로써, b = stra[6].c.unie[3];을 만족하게 된다. |
| c = &(strb[1][2][3].str[2].c.unic);  movq $strb+196376, c(%rip)  struct StrB 타입의 크기는 296byte이다. 따라서 strb[1][2][3].str[2].c.unic의 주소는 strb+296(StrB)\*(1\*20\*30+2\*30+3)+4(int)+1(char)+3(word alignment+56(StrA)\*2+4(int)+1(char)+3 (word alignment)=strb+196376이다. 따라서 주소인 strb+196376를 c로 옮기게 된다. |
| aaa = stra[3].d;  movss stra+216(%rip), %xmm0  movss %xmm0, aaa(%rip)  struct StrA타입의 크기는 56byte이다. 따라서 stra[3].d의 주소는 stra+56(StrA)\*3(인덱스 전 원소의 개수)+4(int)+1(char)+3(word alignment)+40(Uni)=stra+216이다. 따라서 stra+216에 있는 값을 xmm0으로 옮기고, xmm0을 다시 aaa로 옮김으로써 aaa = stra[3].d;를 만족하게 된다. |
| bbb = &(strb[4][6][11].Bd);  movq $strb+767224, bbb(%rip)  struct StrB 타입의 크기는 296byte이다. 따라서 strb[4][6][11].Bd의 주소는 strb+296(StrB)\*(4\*20\*30+6\*30+11)+4(int)+1(char)+3(word alignment)+56(StrA)\*5=strb+767224이다. 따라서 주소인 strb+767224를 bbb로 옮기게 된다. |
| menu1 = strb[9][10][11].str[3].c.menu2;  movl strb+1690640(%rip), %eax  movl %eax, menu1(%rip)  struct StrB 타입의 크기는 296byte이다. 따라서 strb[9][10][11].str[3].c.menu2의 주소는 strb+296(StrB)\*(9\*20\*30+10\*30+11)+4(int)+1(char)+3(word alignment)+56(StrA)\*3+4(int)+1(char)+ 3(word alignment)=strb+1690640이다. 따라서 strb+1690640에 있는 값을 eax로 옮기고, eax를 menu1로 옮김으로써 menu1 = strb[9][10][11].str[3].c.menu2;를 만족하게 된다. |
| cc = &(strb[3][5][6].str[1].b);  movq $strb+579044, cc(%rip)  strb[3][5][6].str[1].b의 주소는 strb+296(StrB)\*(3\*20\*30+5\*30+6)+4(int)+1(char)+3(word alignment)+56(StrA)\*1+4(int)=strb+579044이다. 따라서 주소인 strb+579044를 cc로 옮기게 된다. |
| dd= (strb[3][5][6].str[1].b);  movzbl strb+579044(%rip), %eax  movb %al, dd(%rip)  바로 위에서 구했듯이, strb[3][5][6].str[1].b의 주소는 strb+579044이다. 따라서 strb+579044에 있는 값을 eax로 옮기고, al를 dd로 옮김으로써 dd= (strb[3][5][6].str[1].b);를 만족하게 된다. |
| cc=stra[2].c.unid;  movq stra+120(%rip), %rax  movq %rax, cc(%rip)  struct StrA타입의 크기는 56byte이다. 따라서 stra[2].c.unid의 주소는 stra+56(StrA)\*2(인덱스 전 원소의 개수)+4(int)+1(char)+3(word alignment) = stra+120이다. 따라서 stra+120에 있는 값을 rax로 옮기고, rax를 다시 cc로 옮김으로써 cc=stra[2].c.unid;를 만족하게 된다. |
| strb[5][7][8].Bd=stra[3].d;  movss stra+216(%rip), %xmm0  movss %xmm0, strb+952816(%rip)  먼저 stra[3].d의 주소를 구하게 되면 stra+56(StrA)\*3+4(int)+1(char)+3(word alignment)+40(Uni) =stra+216이다. 그리고 strb[5][7][8].Bd의 주소를 구하게 되면, strb+296(StrB)\*(5\*20\*30+7\*30+8)+4(int)+1(char)+3(word alignment)+56(StrA)\*5=strb+952816이다. 따라서 stra+216에 있는 값을 xmm0로 옮기고, xmm0를 다시 strb+952816으로 옮김으로써 strb[5][7][8].Bd=stra[3].d;를 만족하게 된다. |
| bbb= &(strb[4][6][1].Bd)+2;  movl $strb+764272, %eax  movq %rax, bbb(%rip)  먼저 strb[4][6][1].Bd의 주소는 strb+296(StrB)\*(4\*20\*30+6\*30+1)+4(int)+1(char)+3(word alignment)+56(StrA)\*5==strb+764264이다. strb[4][6][1].Bd의 타입이 float형이기 때문에 뒤쪽의 +2는 +2\*4(float)의 의미를 가지고 있으므로 &(strb[4][6][1].Bd)+2는 strb+764264+2\*4= strb+764272이다. 따라서 strb+764272, 즉 주소를 eax로 옮기고, 다시 rax를 bbb로 옮김으로써, bbb= &(strb[4][6][1].Bd)+2;를 만족하게 된다. |
| cc = &(strb[3][5][6].str[1].b)+1;  movl $strb+579045, %eax  movq %rax, cc(%rip)  위해서 구하였을 때 strb[3][5][6].str[1].b의 주소는 strb+579044였다. strb[3][5][6].str[1].b의 자료형이 char형이므로, 뒤쪽의 +1은 +1\*1(char)의 의미를 가지고 있으므로 &(strb[3][5][6].str[1].b)+1은 strb+579044+1=strb+579045이다. 따라서 strb+579045, 즉 주소를 eax로 옮기고, 다시 rax를 cc로 옮김으로써 cc = &(strb[3][5][6].str[1].b)+1;를 만족하게 된다. |
| aaaa=strc[2].Cb;  movsd strc+256(%rip), %xmm0  movsd %xmm0, aaaa(%rip)  struct StrC타입의 크기는 96byte이다. 따라서 strc[2].Cb의 주소는 strc+96(StrC)\*2(인덱스 전 원소의 개수)+56(Uni2)+4(float)+4(float) = strc+256이다. 따라서 strc+256에 있는 값을 xmm0로 옮기고, xmm0를 다시 aaaa로 옮김으로써 aaaa=strc[2].Cb;를 만족하게 된다. |
| bbbb=&(strc[2].Cb)+2;  movl $strc+272, %eax  movq %rax, bbbb  바로 위에서 구하였을 때 strc[2].Cb의 주소는 strc+256였다. 이때 strc[2].Cb의 자료형이 double형이므로 뒤쪽의 +2는 +2\*8의 의미를 가지고 있게 된다. 따라서 &(strc[2].Cb)+2= strc+256+2\*8=strc+272이고, 주소 값인 strc+272을 eax에 옮기고, 다시 rax를 bbbb에 옮김으로써 bbbb=&(strc[2].Cb)+2;를 만족하게 된다. |
| a=strc[3].Ca.u2a.c.unie[4];  movl strc+312(%rip), %eax  movl %eax, a(%rip)  struct StrC타입의 크기는 96byte이다. 따라서 strc[3].Ca.u2a.c.unie[4]의 주소는 strc+96(strC)\*3(인덱스 전 원소 개수)+4(int)+1(char)+3(word alignment)+ 4(int배열이므로)\*4(index전 원소의 개수)=strc+312이다. 따라서 strc+312에 있는 값을 eax로 옮기고, eax를 다시 a로 옮김으로써 a=strc[3].Ca.u2a.c.unie[4];를 만족하게 된다. |
| aaaaa=strc[2].Cd;  fldt strc+272(%rip)  fstpt aaaaa(%rip)  struct StrC타입의 크기는 96byte이다. 따라서 strc[2].Cd의 주소는 strc+96(strC)\*2+56(Uni2)+4(float)+4(int)+8(double)+8(char\*)=strc+272이다. 따라서 여기에 있는 값을 aaaaa에 넣게 된다. |
| bbbbb=&(strc[2].Cd)+4;  movl $strc+336, %eax  movq %rax, bbbbb(%rip)  바로 위에서 구하였을 때 strc[2].Cd의 주소는 strc+272였다. 이때 strc[2].Cd의 자료형이 long double형이므로 뒤쪽의 +4는 +4\*16의 의미를 가지고 있게 된다. 따라서 &(strc[2].Cd)+4= strc+272+4\*16=strc+272+64=strc+336이고, 주소 값인 strc+336을 eax에 옮기고, 다시 rax를 bbbbb에 옮김으로써 bbbbb=&(strc[2].Cd)+4;를 만족하게 된다. |