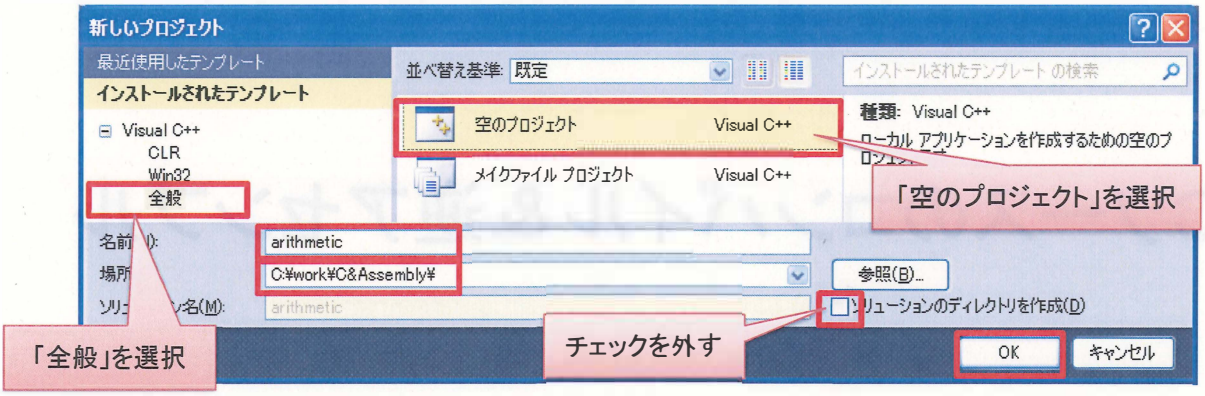
Cソースのコンパイル＆逆アセンブル

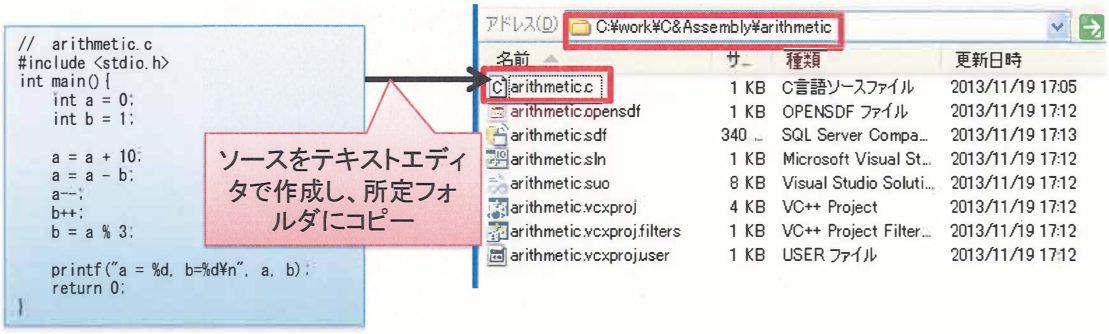
C生成物の逆アセンブル

- いろいろなプログラム構成品から逆アセンブリコードを想像できますか？
- 逆アセンブリコードを高いレベルで理解するためには、ループや条件式のような基本的なプログラム構成品がどのようにコンパイルされるかを理解することが非常に重要
- ここでは、Visual Studio コンパイラを使って検証

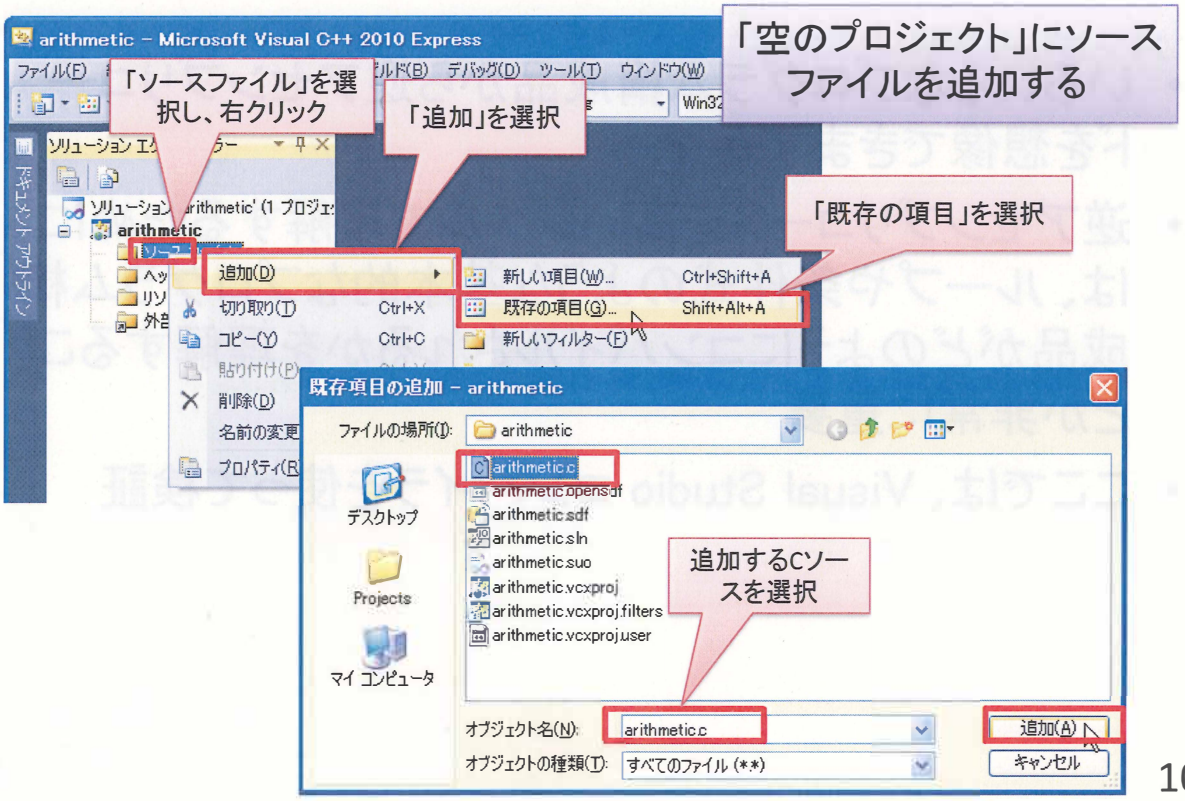
コンパイラの設定(Visual Studio 2010の場合)

- 例：C:¥work¥C&Assemblyにarithmeticプロジェクトを作成

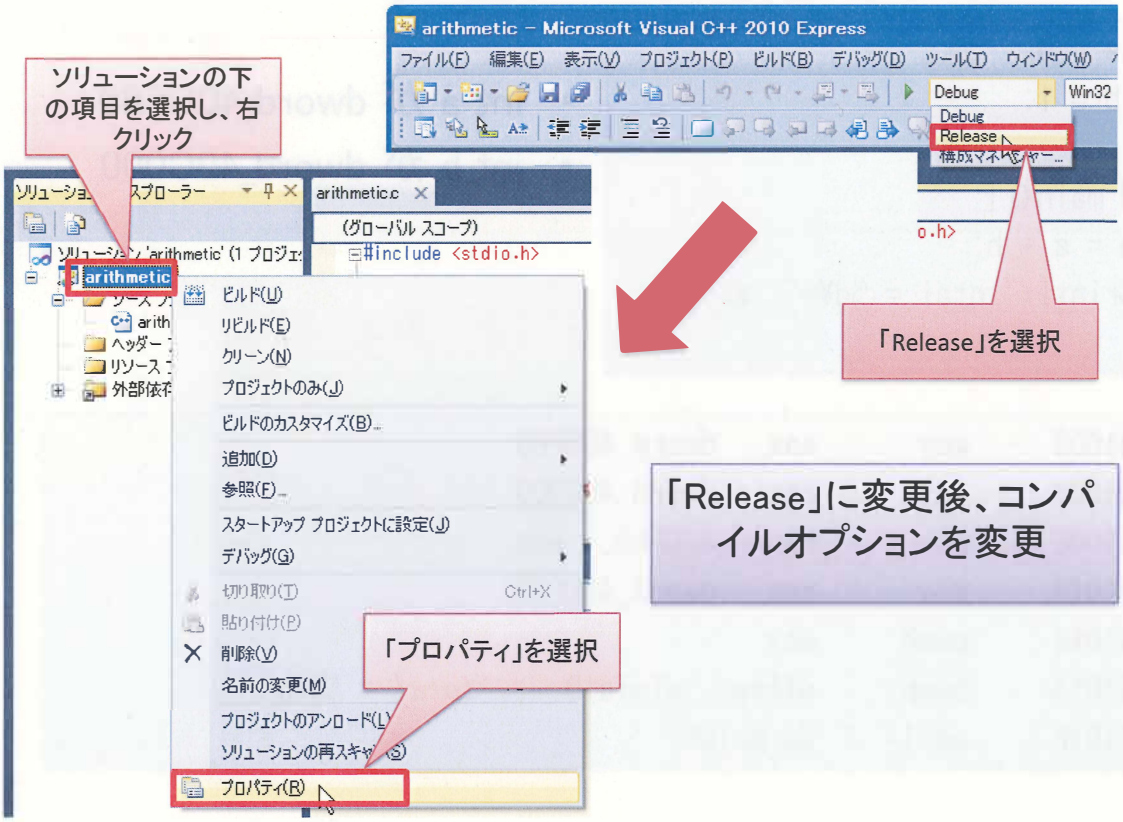




コンパイラの設定(Visual Studio 2010の場合)

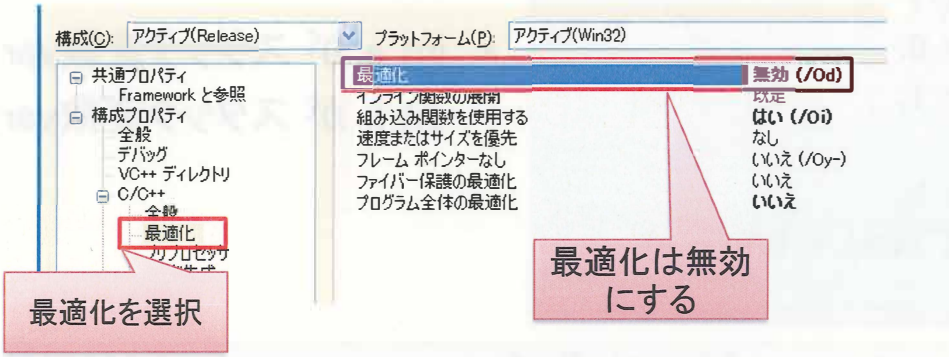


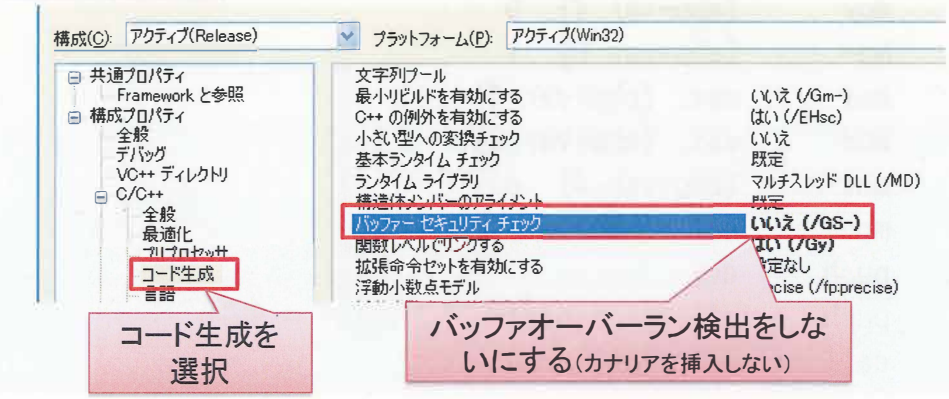
コンパイラの設定(Visual Studio 2010の場合)



コンパイラの設定(Visual Studio 2010の場合)

- シンプルなコードにするため2箇所変更する





グローバル変数(C & Assembly)

```
int a = 0;
int b = 1;

void main(){
   a = a + b;
   printf("total = %d¥n", a);
}
```

- int a が dword_40CF60
- int b が dword_40C000

```
00401003     mov      eax,  dword_40CF60
00401008     add      eax,  dword_40C000
0040100E     mov      dword_40CF60,  eax
00401013     mov      ecx,  dword_40CF60
00401019     push     ecx
0040101A     push     offset  aToTalD   ;"total = %d¥n"
0040101F     call     ds:printf
```


ローカル変数(C & Assembly)

```
void main(){
   int a = 0;
   int b = 1;

   a = a + b;
   printf("total = %d¥n", a);
}
```

- int a が スタック変数var_4
- int b が スタック変数var_8

```
00401006     mov      [ebp+var_4], 0
0040100D     mov      [ebp+var_8], 1
00401014     mov      eax, [ebp+var_4]
00401017     add      eax, [ebp+var_8]
0040101A     mov      [ebp+var_4], eax
0040101D     mov      ecx, [ebp+var_4]
00401020     push     ecx
00401021     push     offset  aToTalD   ;"total = %d¥n"
00401026     call     ds:printf
```

演算(C & Assembly)

```
int a = 0;
int b = 1;
a = a + 10;
a = a - b;
a--;
b++;
b = a % 3;
```

- a がvar_4
- b がvar_8

```
00401006    mov     [ebp+var_4], 0
0040100D    mov     [ebp+var_8], 1
00401014    mov     eax, [ebp+var_4]
00401017    add     eax, 0Ah
0040101A    mov     [ebp+var_4], eax
0040101D    mov     ecx, [ebp+var_4]
00401020    sub     ecx, [ebp+var_8]
00401023    mov     [ebp+var_4], ecx
00401026    mov     edx, [ebp+var_4]
00401029    sub     edx, 1
0040102C    mov     [ebp+var_4], edx
0040102F    mov     eax, [ebp+var_8]
00401032    add     eax, 1
00401035    mov     [ebp+var_8], eax
00401038    mov     eax, [ebp+var_4]
0040103B    cdq
0040103C    mov     ecx, 3
00401041    idiv    ecx
00401043    mov     [ebp+var_8], edx
```


If文(C & Assembly)

```
int a = 0;
int b = 1;
int(a==b)
   printf("a equals b¥n");
else
   printf("a not equals to b¥n");
}
```

- int a が var_4
- int b が var_8

```
00401006    mov     [ebp+var_4], 0
0040100D    mov     [ebp+var_8], 1
00401014    mov     eax, [ebp+var_4]
00401017    cmp     eax, [ebp+var_8]
0040101A    jnz     short loc_40102C
0040101C    push    offset  Format    ;"a equals b¥n"
00401021    call    ds:printf
00401027    add     esp, 4
0040102A    jmp     short loc_40103A
0040102C loc_40102C:
0040102C    push    offset  aANotEqualToB    ;"a not equals to b¥n"
00401031    call    ds:printf
```

ネストしたIf文(C & Assembly)

```
int a = 0;
int b = 1;
int c = -1;

if(a == b){
  if(c==0){
    printf("c zero and a = b¥n");
  }else{
    printf("c non-zero and a = b¥n");
  }
}else{
  if(c==0){
    printf("c zero and a not= b¥n");
  }else{
    printf("c non-zero and a not= b¥n");
  }
}
```

```
00401006     mov     [ebp+var_4], 0
0040100D     mov     [ebp+var_8], 1
00401014     mov     [ebp+var_C], 0FFFFFFFFh
0040101B     mov     eax, [ebp+var_4]
0040101E     cmp     eax, [ebp+var_8]
00401021     jnz     short loc_401049
00401023     cmp     [ebp+var_C], 0
00401027     jnz     short loc_401039
00401029     push    offset Format ;"c zero and a = b¥n"
0040102E     call    ds:printf
00401034     add     esp, 4
00401037     jmp     short loc_00401047
00401039 loc_401039:
00401039     push    offset aCNonZeroAndAB ;"c non-zero and a = b¥n"
0040103E     call    ds:printf
00401044     add     esp, 4
00401047 loc_401047:
00401047     jmp     short loc_40106D
00401049 loc_401049:
00401049     cmp     [ebp+var_C], 0
0040104D     jnz     short loc_40105F
0040104F     push    offset aCZeroAndANotB ;"c zero and a not= b¥n"
00401054     call    ds:printf
0040105A     add     esp, 4
0040105D     jmp     short loc_40106D
0040105F loc_40105F:
0040105F     push    offset aCNonZeroAndANo ;"c non-zero and a not= b¥n"
00401064     call    ds:printf
```


関数コール(C & Assembly)

```
int simple_function(int a, int b){
  return a+b;
}
int main(){
  int a = 1;
  int b = 2;

  printf("the function returned %d¥n",
    simple_function(a,b));
}
```

simple_function()

```
00401000 arg_0 = dword ptr 8
00401000 arg_4 = dword ptr 0Ch
00401000
00401000     push    ebp
00401001     mov     ebp, esp
00401003     mov     eax, [ebp+arg_0]
00401006     add     eax, [ebp+arg_4]
00401009     pop     ebp
00401007     retn
```

main()

```
00401016     mov     [ebp+var_4], 1
0040101D     mov     [ebp+var_8], 2
00401024     mov     eax, [ebp+arg_8]
00401027     push    eax
00401028     mov     ecx, [ebp+var_4]
0040102B     push    ecx
0040102C     call    sub_401000
00401031     add     esp,8
00401034     push    eax
00401035     push    offset Format; "the function returned %d¥n"
0040103A     call    ds:printf
```