vc-5. インラインアセンブラ
（Visual Studio C++ の機能と操作）

https://www.kkaneko.jp/cc/vc/index.html

金子邦彦

目次

5-1 インラインアセンブラ

インラインアセンブラ

- インラインアセンブラとは、
- 他の言語の中に、アセンブリ言語のプログラムを埋め込むこと

```
int main()
{
    int a;
    _asm {
        mov a, 100
    }
    printf("a = %d", a);

    return 0;
}
```

アセンブリ言語のプログラムの埋め込み

パソコン演習

- Visual Studio 2015 を起動しなさい
- Visual Studio 2015 で、Win32 コンソールアプリケーション用プロジェクトを新規作成しなさい

プロジェクトの「名前」は何でもよい

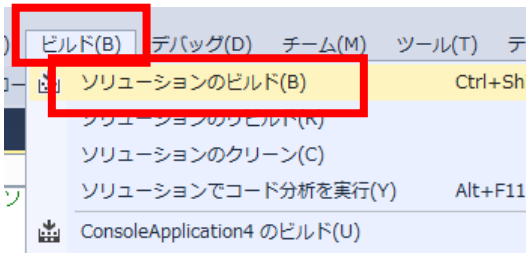
- Visual Studio 2015 のエディタを使って、ソースファイルを編集しなさい

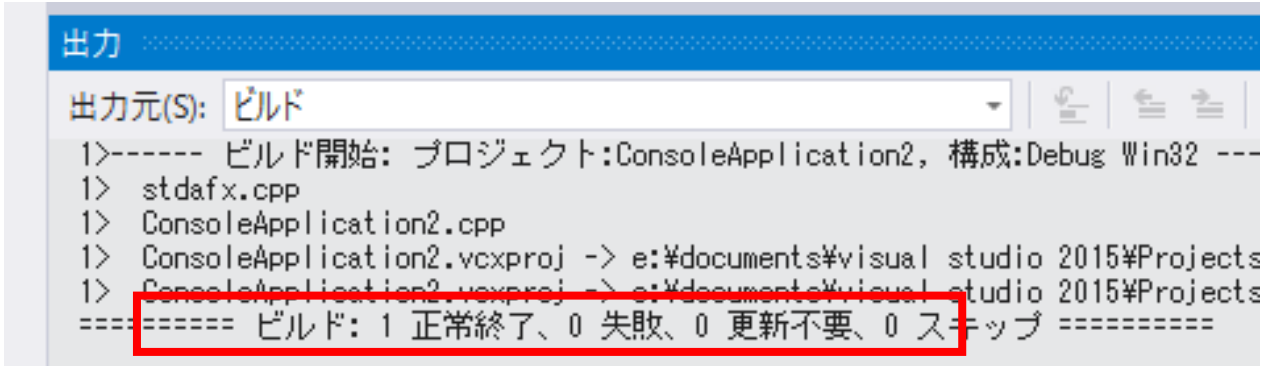
```
int main()
{
    int a;
    _asm {
        mov a, 100
    }
    printf("a = %d", a);

    return 0;
}
```

5 行追加

- ビルドしなさい．ビルドのあと「１　正常終了、０　失敗」の表示を確認しなさい

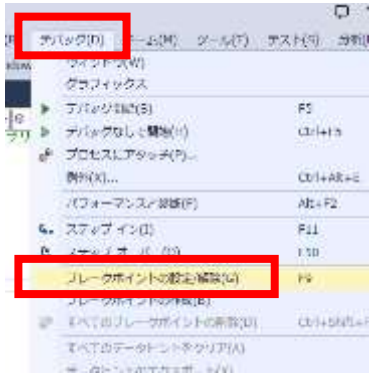
→　表示されなければ、プログラムのミスを自分で確認し、修正して、ビルドをやり直す





- Visual Studio 2015 で「printf」の行に、ブレークポイントを設定しなさい

```
int main()
{
    int a;
    _asm {
        mov a, 100
    }
    printf("a = %d", a);

    return 0;
}
```



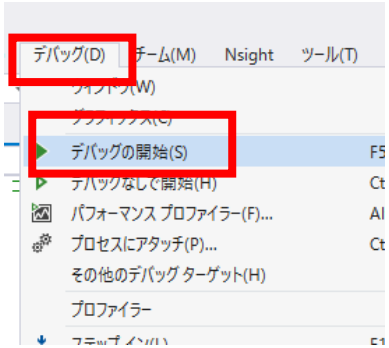

```
 7  int main()
 8  {
 9      int a;
10      _asm {
11          mov a, 100
12      }
13      printf("a = %d", a);
14
15      return 0;
16  }
17
```

① 「**printf**」の行をマウスでクリック

② 「デバッグ」→「**ブレークポイントの設定/解除**」

③ ブレークポイントが設定されるので確認．赤丸がブレークポイントの印

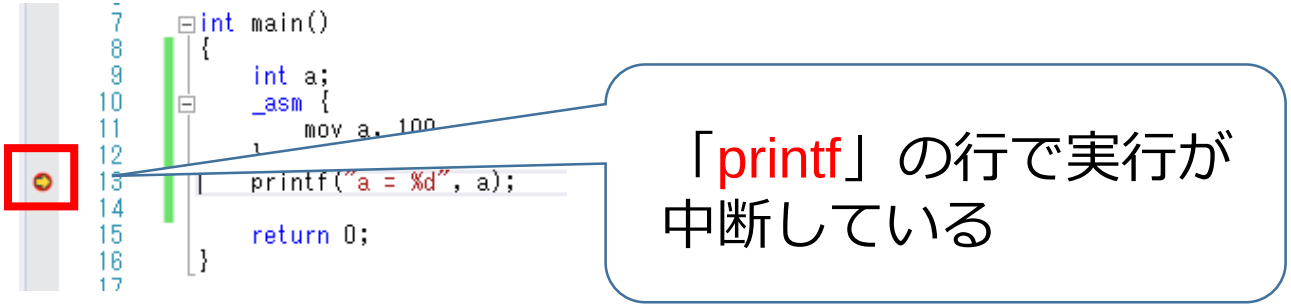
- Visual Studio 2015 で、デバッガーを起動しなさい.



「デバッグ」
→「**デバッグの開始**」

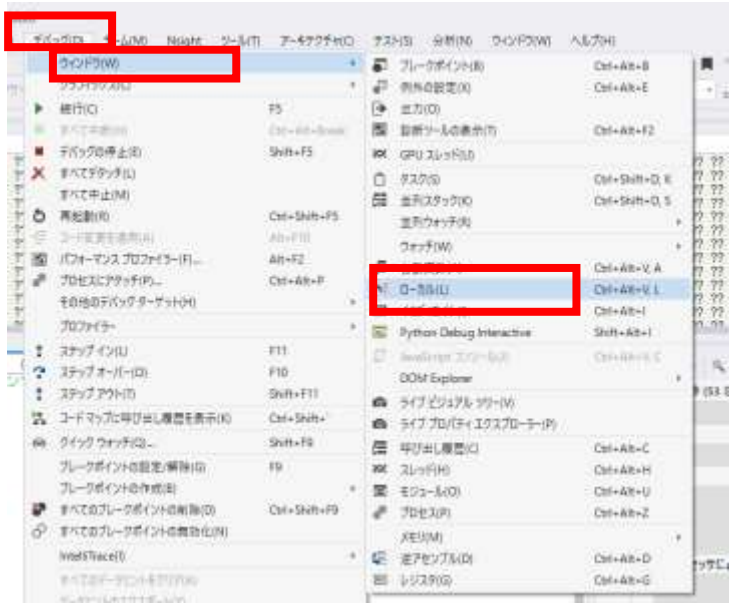
- 「printf」の行で、実行が中断することを確認しなさい

あとで使うので、中断したままにしておくこと

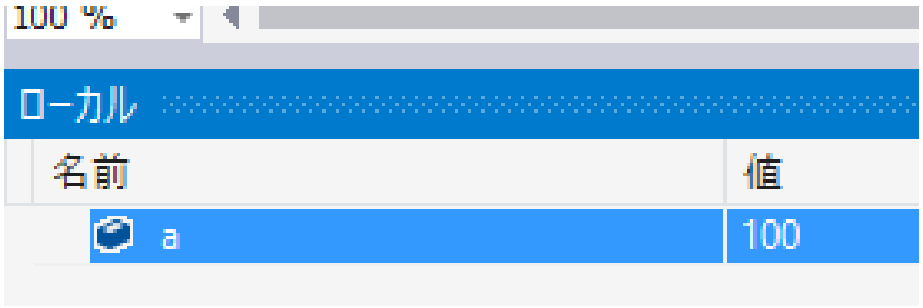


「printf」の行で実行が中断している

- 「printf」の行で、実行が中断した状態で、変数の値を表示させなさい．手順は次の通り．

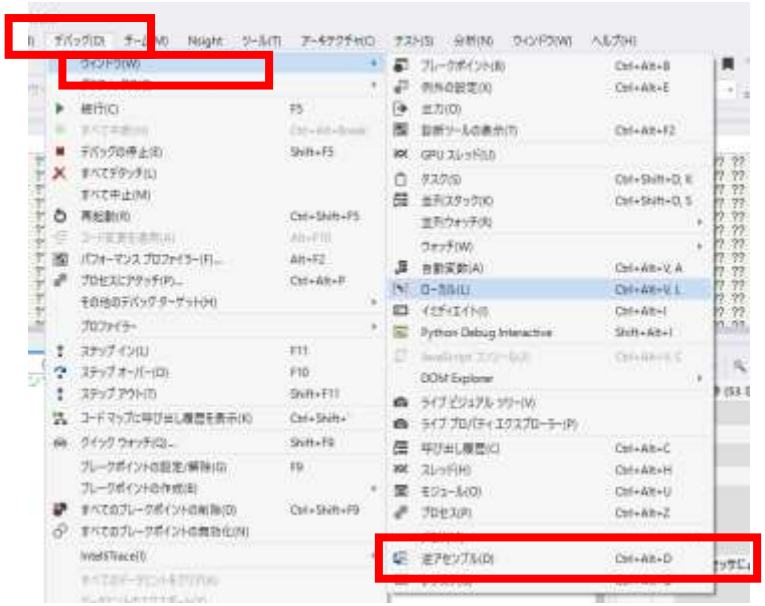


①「デバッグ」
→「ウインドウ」
→ 「**ローカル**」



② 変数名と値の対応表が表示される

- 「printf」の行で、実行が中断した状態で、逆アセンブルを行いなさい.



```
009F1A9A  push        esi
009F1A9B  push        edi
009F1A9C  lea         edi,[ebp-0D0h]
009F1AA2  mov         ecx,34h
009F1AA7  mov         eax,0CCCCCCCCh
009F1AAC  rep stos    dword ptr es:[edi]
009F1AAE  mov         eax,dword ptr [__security_cookie (09F9024h)]
009F1AB3  xor         eax,ebp
009F1AB5  mov         dword ptr [ebp-4],eax
    int a;
    _asm {
        mov a, 100
009F1AB8  mov         dword ptr [a],64h
    }
    printf("a = %d", a);
009F1ABF  mov         eax,dword ptr [a]
009F1AC2  push        eax
009F1AC3  push        offset string "a = %d" (09F6B94h)
009F1AC8  call        _printf (09F134Dh)
009F1ACD  add         esp,8

    return 0;
009F1AD0  xor         eax,eax
```

① 「デバッグ」→「ウインドウ」→「**逆アセンブル**」

② 逆アセンブルの結果が表示される

- 1次のことを確認しなさい（Visual Studioでの操作ではない）．

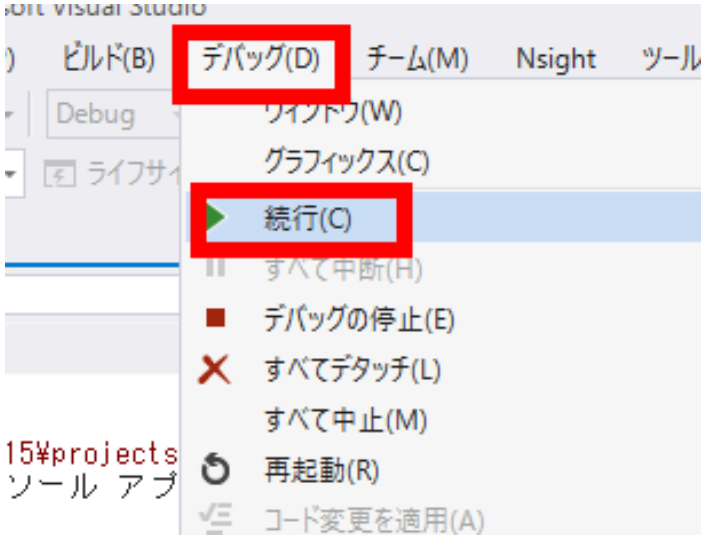
```
009F1A9A  push        esi
009F1A9B  push        edi
009F1A9C  lea         edi,[ebp-0D0h]
009F1AA2  mov         ecx,34h
009F1AA7  mov         eax,0CCCCCCCCh
009F1AAC  rep stos    dword ptr es:[edi]
009F1AAE  mov         eax,dword ptr [__security_cookie (09F9024h)]
009F1AB3  xor         eax,ebp
009F1AB5  mov         dword ptr [ebp-4],eax
    int a;
    _asm {
        mov a, 100
009F1AB8  mov         dword ptr [a],64h
    }
    printf("a = %d", a);
009F1ABF  mov         eax,dword ptr [a]
009F1AC2  push        eax
009F1AC3  push        offset string "a = %d" (09F6B94h)
009F1AC8  call        _printf (09F134Dh)
009F1ACD  add         esp,8

    return 0;
009F1AD0  xor         eax,eax
```

・Visual C++ の変数「a」は「dword ptr [a]」と翻訳され、

・10進数の「100」は16進数の「64」に翻訳されている

mov a, 100
は
move dword ptr [a],64h
に変化
※ どちらも**同じ**意味

- 最後に、プログラム実行の再開の操作を行いなさい。これで、デバッガーが終了する。



「デバッグ」
→「**続行**」

5-2 算術演算命令の例

算術演算の例

```
int main()
{
    int a;
    _asm {
        mov a, 100;
        add a, 200;
    }
    printf("a = %d", a);
    return 0;
}
```

アセンブリ言語のプログラム

a に 100 をセット

a に 200 を足しこむ

パソコン演習

- Visual Studio 2015 を起動しなさい
- Visual Studio 2015 で、Win32 コンソールアプリケーション用プロジェクトを新規作成しなさい

プロジェクトの「名前」は何でもよい

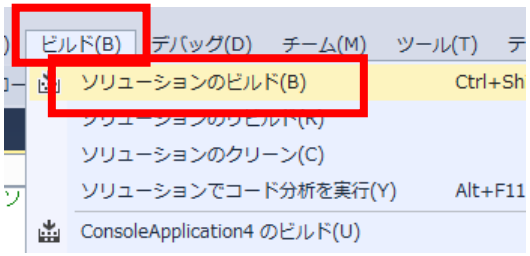
- Visual Studio 2015 のエディタを使って、ソースファイルを編集しなさい

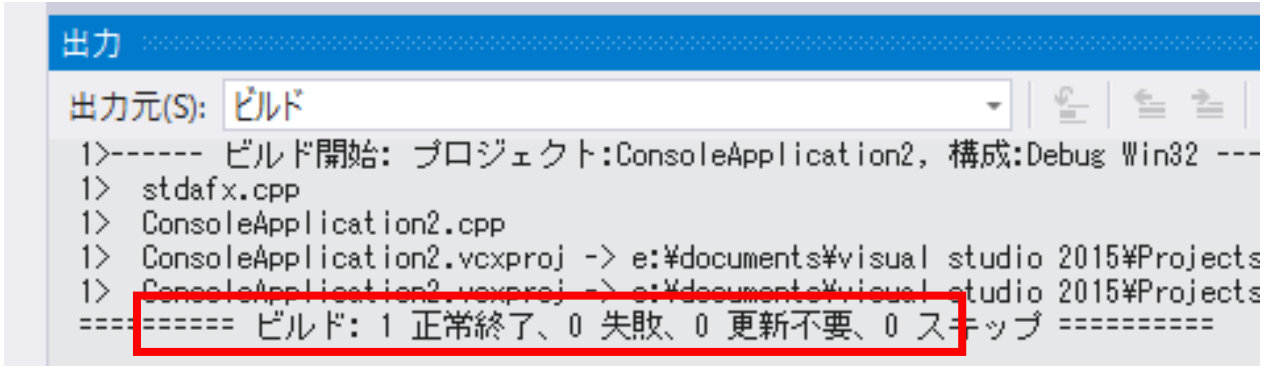
```
int main()
{
    int a;
    _asm {
        mov a, 100;
        add a, 200;
    }
    printf("a = %d", a);
    return 0;
}
```

６行追加

- ビルドしなさい．ビルドのあと「１　正常終了、０　失敗」の表示を確認しなさい

→　表示されなければ、プログラムのミスを自分で確認し、修正して、ビルドをやり直す

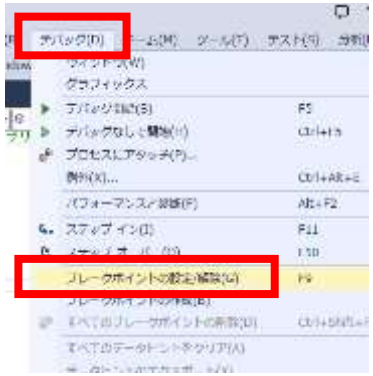




- Visual Studio 2015 で「printf」の行に、ブレークポイントを設定しなさい

```
int main()
{
    int a;
    _asm {
        mov a, 100;
        add a, 200;
    }
    printf( a = %d , a);
    return 0;
}
```

① 「**printf**」の行をマウスでクリック

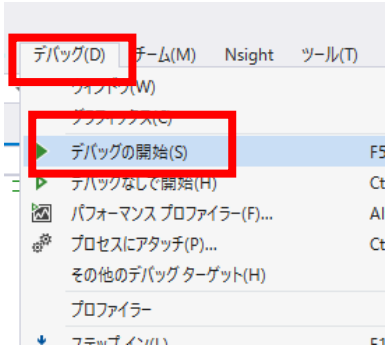


② 「デバッグ」→「**ブレークポイントの設定/解除**」

```
7   int main()
8   {
9       int a;
10      _asm {
11          mov a, 100;
12          add a, 200;
13      }
14      printf("a = %d", a);
15      return 0;
16  }
```

③ ブレークポイントが設定されるので確認．赤丸がブレークポイントの印

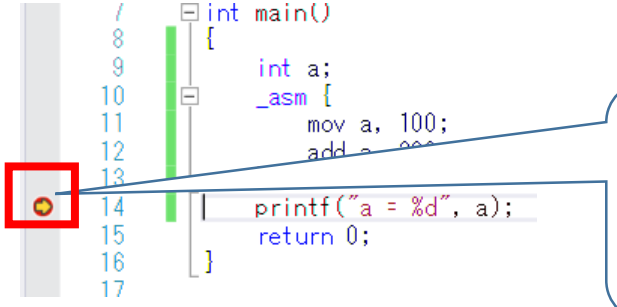
- Visual Studio 2015 で、デバッガーを起動しなさい.



「デバッグ」
→「**デバッグの開始**」

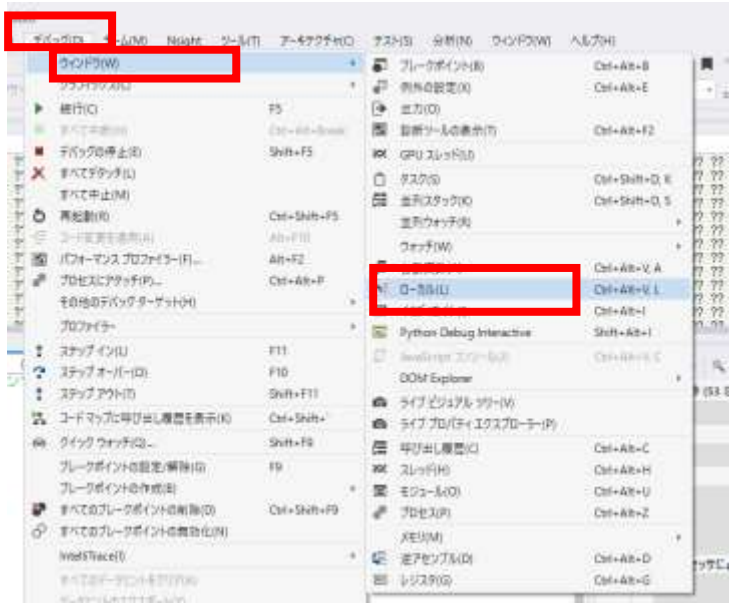
- 「printf」の行で、実行が中断することを確認しなさい

あとで使うので、中断したままにしておくこと

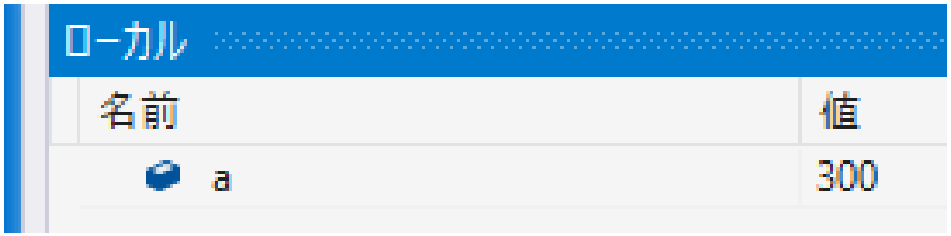


「printf」の行で実行が中断している

- 「printf」の行で、実行が中断した状態で、変数の値を表示させなさい．手順は次の通り．

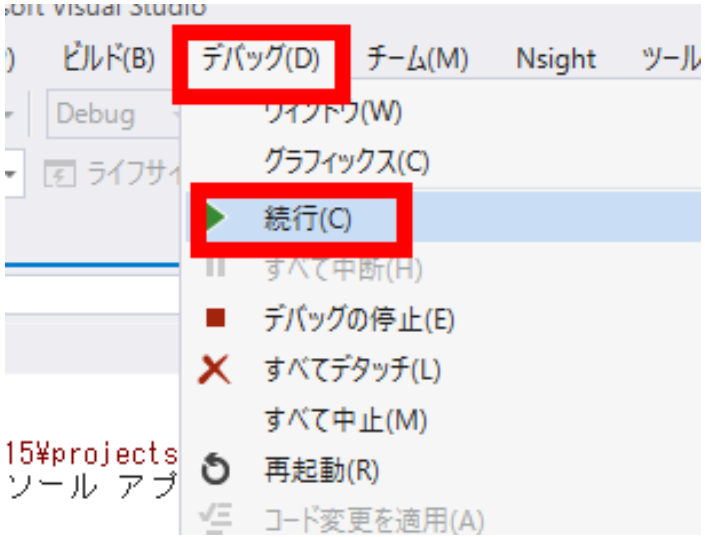


①「デバッグ」
→「ウインドウ」
→ 「**ローカル**」



② 変数名と値の対応表が表示される

- 最後に、プログラム実行の再開の操作を行いなさい。これで、デバッガーが終了する。



「デバッグ」
→「**続行**」

- 次のように書き替えて、同じ手順を繰り返しなさい.
- そして，変数 a の値を確認しなさい

```
int main()
{
    int a;
    _asm {
        mov a, 30;
        add a, 20;
    }
    printf("a = %d", a);
    return 0;
}
```

add 加算

- 次のように書き替えて、同じ手順を繰り返しなさい.
- そして，変数 a の値を確認しなさい

```
int main()
{
    int a;
    _asm {
        mov a, 30;
        sub a, 20;
    }
    printf("a = %d", a);
    return 0;
}
```

sub 加算

- 書き替えて、同じ手順を繰り返しなさい.
- そして，変数 a の値を確認しなさい

```
int main()
{
    int a;
    _asm {
        mov a, 30;
        imul eax, a, 20;
        mov a, eax;
    }
    printf("a = %d", a);
    return 0;
}
```

imul 乗算
次ページに解説

```
int main()
{
    int a;
    _asm {
        mov a, 30;
        imul eax, a, 20;
        mov a, eax;
    }
    printf("a = %d", a);
    return 0;
}
```

アセンブリ言語のプログラム

① a に 30 をセット

② a × 20 の結果を、**レジスタ** eax にセット

③ a に**レジスタ** eax の値をセット