

# ca-12. スタック

(コンピュータ・アーキテクチャ演習)

URL: <https://www.kkaneko.jp/cc/ca/index.html>

金子邦彦



# スタックのプッシュとポップ



- プッシュ (push) :  
スタックの一番上に追加
- ポップ (pop) :  
スタックの一番上から削除

**スタック**は,  
複数のデータを格納できるデータ構造

- Visual Studio を起動しなさい
- Visual Studio で, Win32 コンソールアプリケーション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

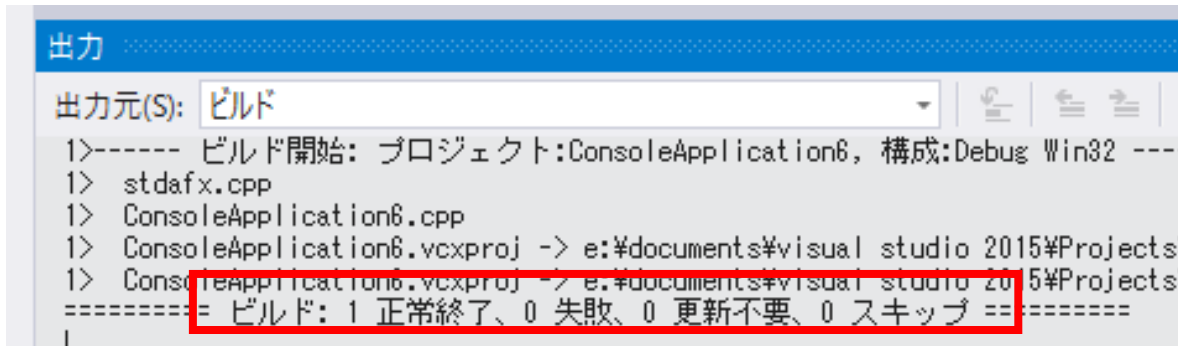
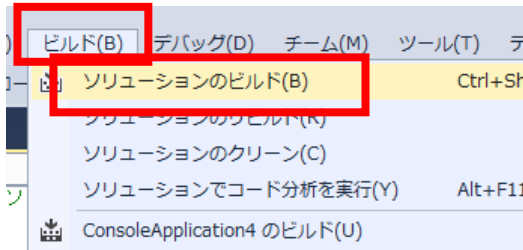
- Visual Studioのエディタを使って、ソースファイルを編集しなさい

```
int main()  
{  
    _asm {  
        push 10;  
        push 5;  
        push 20;  
    }  
    return 0;  
}
```

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

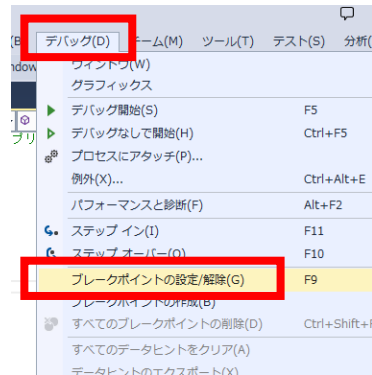
→ 表示されなければ、プログラムのミスを自分で確認し、修正して、ビルドをやり直す



A screenshot of the Visual Studio Output window. The title bar is '出力'. The '出力元(S):' is set to 'ビルド'. The output text shows the start of a build for 'プロジェクト:ConsoleApplication6, 構成:Debug Win32'. The final line of the output is highlighted with a red box: '=====  
= ビルド: 1 正常終了, 0 失敗, 0 更新不要, 0 スキップ ==  
====='.

- Visual Studioで「push 10;」の行に、ブレークポイントを設定しなさい

```
int main()
{
    _asm {
        push 10;
        push 5;
        push 20;
    }
    return 0;
}
```



```
7
8
9
10
11
12
13
14
15
16
```

```
int main()
{
    _asm {
        push 10;
        push 5;
        push 20;
    }
    return 0;
}
```

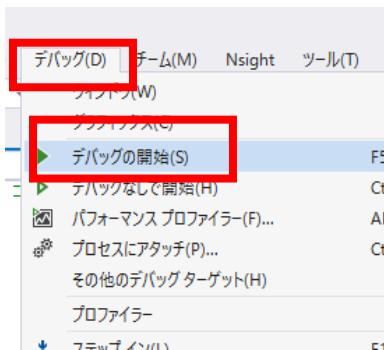


① 「push 10;」の行をマウスでクリック

② 「デバッグ」→「ブレークポイントの設定/解除」

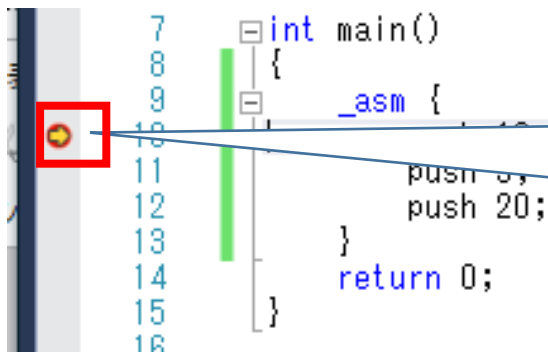
③ ブレークポイントが設定されるので確認。  
赤丸がブレークポイントの印

- Visual Studioで、デバッガーを起動しなさい。



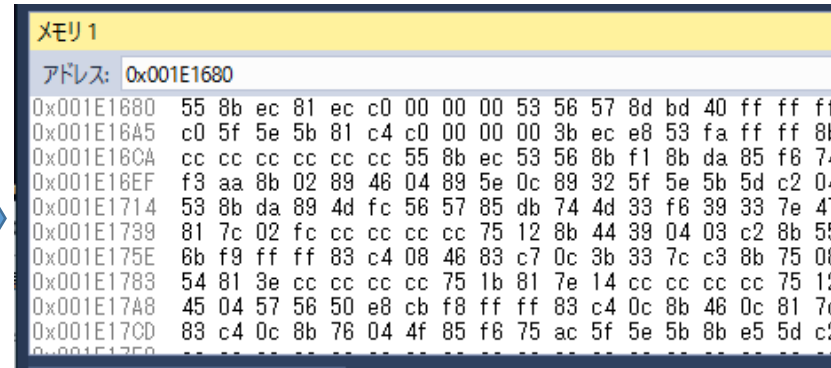
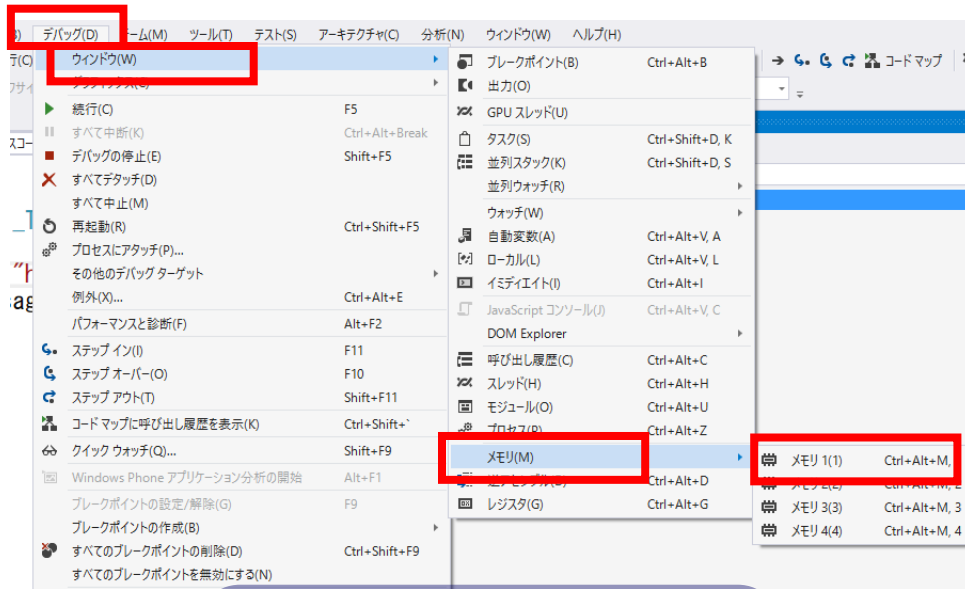
「デバッグ」  
→ 「デバッグ開始」

- 「push 10;」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



「push 10;」の行で実行が  
中断している

- 「push 10;」 の行で，実行が中断した状態で，メモリの中身を表示させなさい．手順は次の通り．



② 「メモリ 1」の画面が表示される

① 「デバッグ」  
→ 「ウィンドウ」  
→ 「メモリ」  
→ 「メモリ 1 (1)」



- 「メモリ1」の画面の「アドレス」に「esp - 12」と入れて Enter キーを押さない

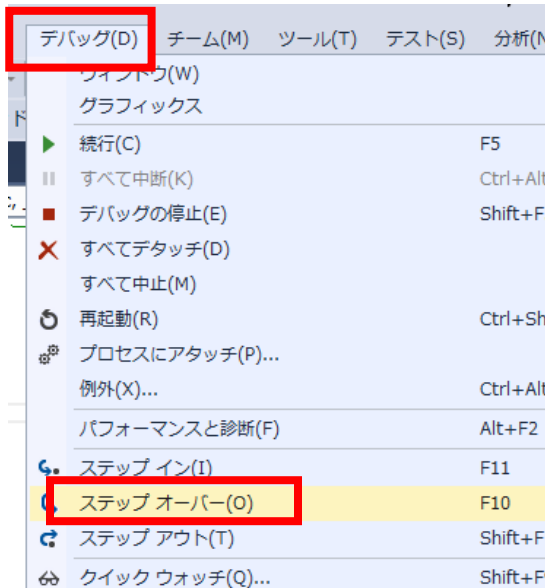
| メモリ        |   |
|------------|---|
| アドレス:      | esp - 12                                    |
| 0x0133FA48 | ?? ?? ?? ?? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? |
| 0x0133FA6D | ?? ?? ?? ?? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? |
| 0x0133FA92 | ?? ?? ?? ?? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? |
| 0x0133FAB7 | ?? ?? ?? ?? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? |
| 0x0133FADC | ?? ?? ?? ?? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? |
| 0x0133FB01 | ?? ?? ?? ?? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? |
| 0x0133FB26 | ?? ?? ?? ?? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? |
| 0x0133FB4B | ?? ?? ?? ?? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? |
| 0x0133FB70 | ?? ?? ?? ?? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? |
| 0x0133FB95 | ?? ?? ?? ?? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? |
| 0x0133FBA  | ?? ?? ?? ?? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? |



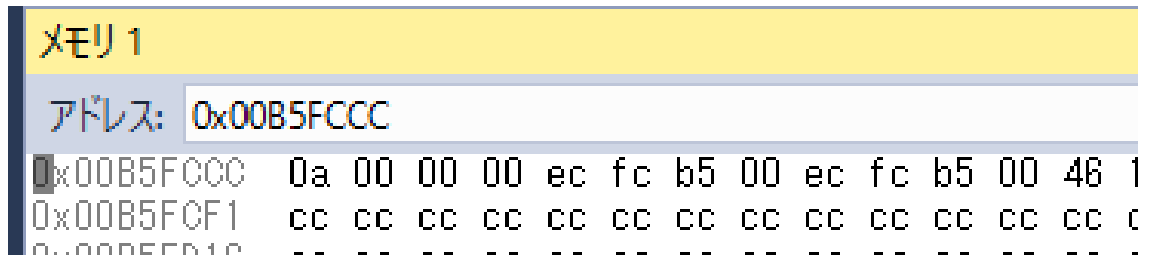
| メモリ        |   |
|------------|---|
| アドレス:      | 0x00B5FCCC  |
| 0x00B5FCCC | 0a 00 00 00 ec fc b5 00 ec fc b5 00 48 10 1e 00 46 10 |
| 0x00B5FCF1 | cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |
| 0x00B5FD16 | cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |
| 0x00B5FD3B | cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |
| 0x00B5FD60 | cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |
| 0x00B5FD85 | cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc |
| 0x00B5FDAA | 1e 00 01 00 00 00 b0 93 da 02 38 a8 da 02 10 fe b5 00 |
| 0x00B5FDCF | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    |
| 0x00B5FDF4 | 00 00 00 00 c0 fd b5 00 00 00 00 00 6c fe b5 00 60 37 |
| 0x00B5FE19 | fe b5 00 98 1d 1e 00 34 fe b5 00 74 86 01 77 00 d0 8a |
| 0x00B5FE2F | 0- 00 10 4b 1d 0- 00 00 00 00 00 00 00 00 10 0-       |

**Enter キーを押すと  
画面が変化するので確認する**

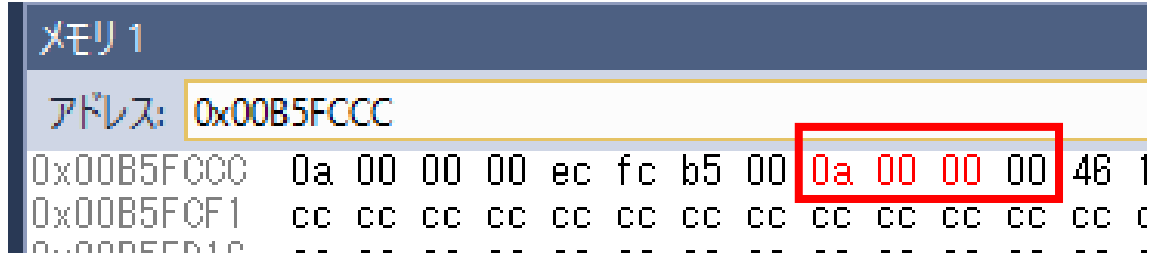
- ステップオーバーの操作を1回行い，変化を見なさい



「デバッグ」  
→ 「ステップオーバー」  
(あるいは F10 キー)



↓ push 10 (16 進数で 0A)



- ステップオーバーの操作を，さらに2回行い，変化を見なさい

```
メモリ  
アドレス: 0x00B5FCCC  
0x00B5FCCC 0a 00 00 00 ec fc b5 00 0a 00 00 00 48 1  
0x00B5FCF1 cc cc cc cc cc cc cc cc cc cc cc cc cc cc c  
0x00B5FD10 .. .. .. .. .. .. .. .. .. .. .. .. .. ..
```

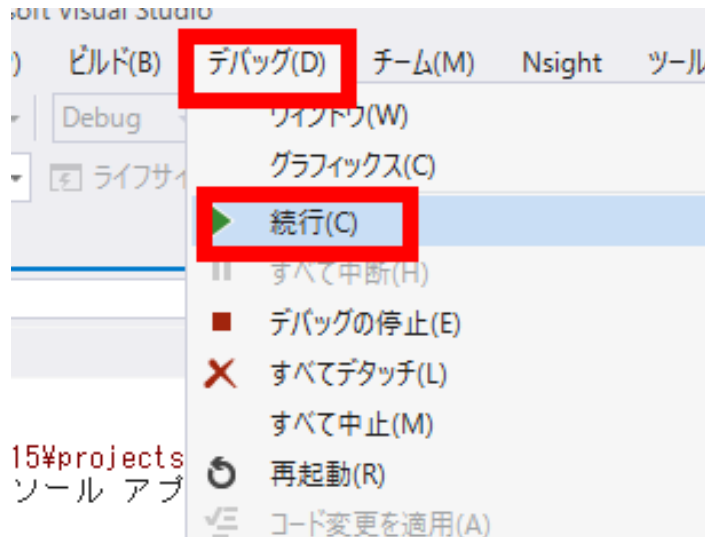
↓ push 5 (16進数で 5)

```
メモリ  
アドレス: 0x00B5FCCC  
0x00B5FCCC 0a 00 00 00 05 00 00 00 0a 00 00 00 48 1  
0x00B5FCF1 cc cc cc cc cc cc cc cc cc cc cc cc cc cc c  
0x00B5FD10 .. .. .. .. .. .. .. .. .. .. .. .. .. ..
```

↓ push 20 (16進数で 14)

```
メモリ  
アドレス: 0x00B5FCCC  
0x00B5FCCC 14 00 00 00 05 00 00 00 0a 00 00 00 48 1  
0x00B5FCF1 cc cc cc cc cc cc cc cc cc cc cc cc cc cc c  
0x00B5FD10 .. .. .. .. .. .. .. .. .. .. .. .. .. ..
```

- 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，同じ手順を繰り返さない。

```
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20
```

```
int main()  
{  
    int a;  
    _asm {  
        push 1;  
        push 2;  
        pop a;  
        push 3;  
        pop a;  
        pop a;  
    }  
    return 0;  
}
```



古いデータが消えずに残っている

