

# ca-11. 条件分岐, 繰り返し

(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



謝辞: 「いらすとや」のイラストを使用しています

# 11-1 比較命令

# 比較命令



- **比較命令**は、何かと何かの比較を行い、フラグレジスタを変化させるための命令

- **条件ジャンプ命令**は、フラグの値がある特定の条件のときだけジャンプする命令

# 比較命令



比較命令は、何かと、何かの比較

```
mov     dword ptr ds:[868130h],0Ah
cmp     dword ptr ds:[868130h],0Ch
jll
mov
jmp
mov     dword ptr ds:[868134h],1F4h
```

変数 age のメモリアドレス 10 進数で 12

変数 age の値をメモリから読み込んで  
12 と比較

# 比較命令の前後でのフラグレジスタの変化



```
0125139E mov     dword ptr ds:[1258130h],14h
      if (age >= 12)
012513A8 cmp     dword ptr ds:[1258130h],0Ch
012513AF jl     wmain+3Dh (012513BDh)
      p = 1800;
012513B1 mov     dword ptr ds:[1258134h],708h
      else
012513BB jmp     wmain+47h (012513C7h)
      p = 500;
012513BD mov     dword ptr ds:[1258134h],1F4h
      return 0;
```

```
00 ECX = 00000000 EDX = 00000001 ESI = 00000000
13A8 ESP = 0043FD70 EBP = 0043FE3C EFL = 00000202
```

比較命令 `cmp` の実行直前



```
0125139E mov     dword ptr ds:[1258130h],14h
      if (age >= 12)
012513A8 cmp     dword ptr ds:[1258130h],0Ch
012513AF jl     wmain+3Dh (012513BDh)
      p = 1800;
012513B1 mov     dword ptr ds:[1258134h],708h
      else
012513BB jmp     wmain+47h (012513C7h)
      p = 500;
012513BD mov     dword ptr ds:[1258134h],1F4h
      return 0;
```

```
00 ECX = 00000000 EDX = 00000001 ESI = 00000000
13AF ESP = 0043FD70 EBP = 0043FE3C EFL = 00000212
```

比較命令 `cmp` の実行直後

# フラグ変化



age = 20 のとき

age = 10 のとき

age と 12 の比較によるフラグの変化

- ◆ ZF (ゼロフラグ) クリア  
age の値は 20 と等しくない
- ◆ SF (サインフラグ) クリア  
age の値は 12 より小さくない

- ◆ ZF (ゼロフラグ) クリア  
age の値は 20 と等しくない
- ◆ SF (サインフラグ) **セット**  
age の値は 12 より小さい



# 11-2 条件分岐とジャンプ命令

# 条件分岐の例



Visual C++ の  
プログラム

```
age = 20;
```

```
if (age >= 12)
```

```
    p = 1800;
```

← こちらが有効

```
else
```

```
    p = 500;
```

← 無視される

```
age = 10;
```

```
if (age >= 12)
```

```
    p = 1800;
```

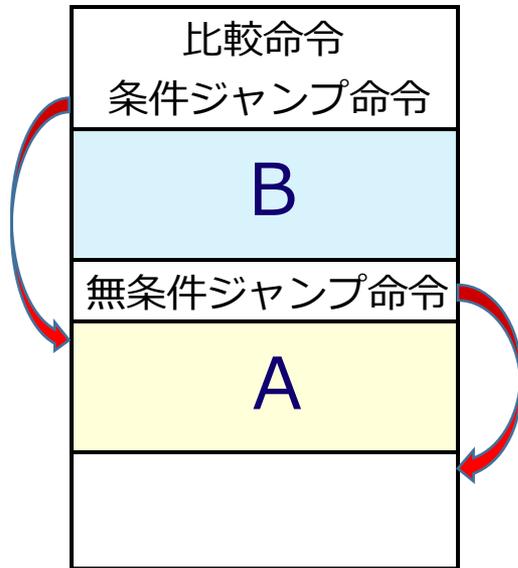
← 無視される

```
else
```

```
    p = 500;
```

← こちらが有効

# 条件分岐でのプログラムの配置

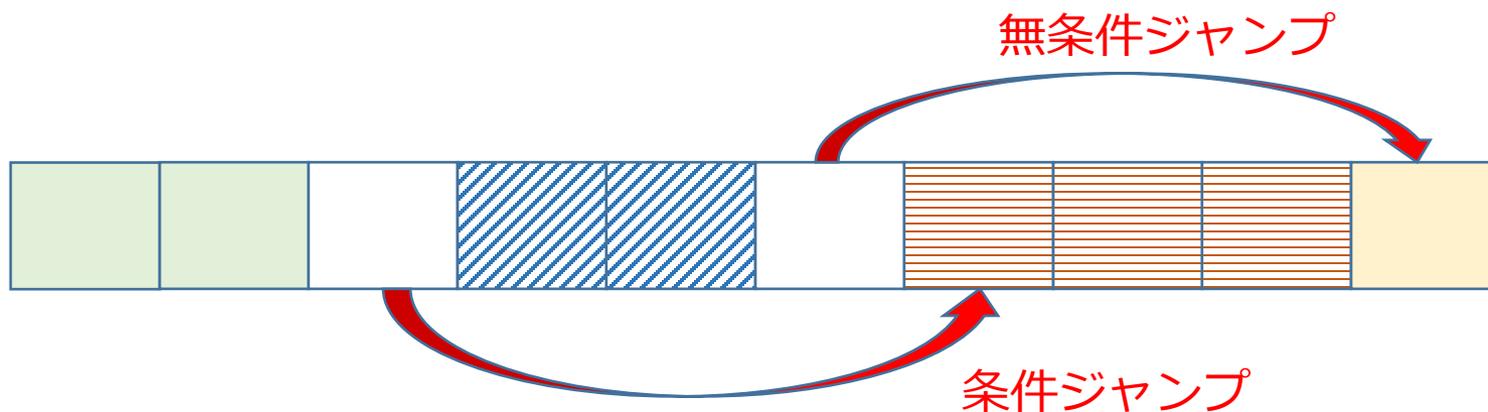


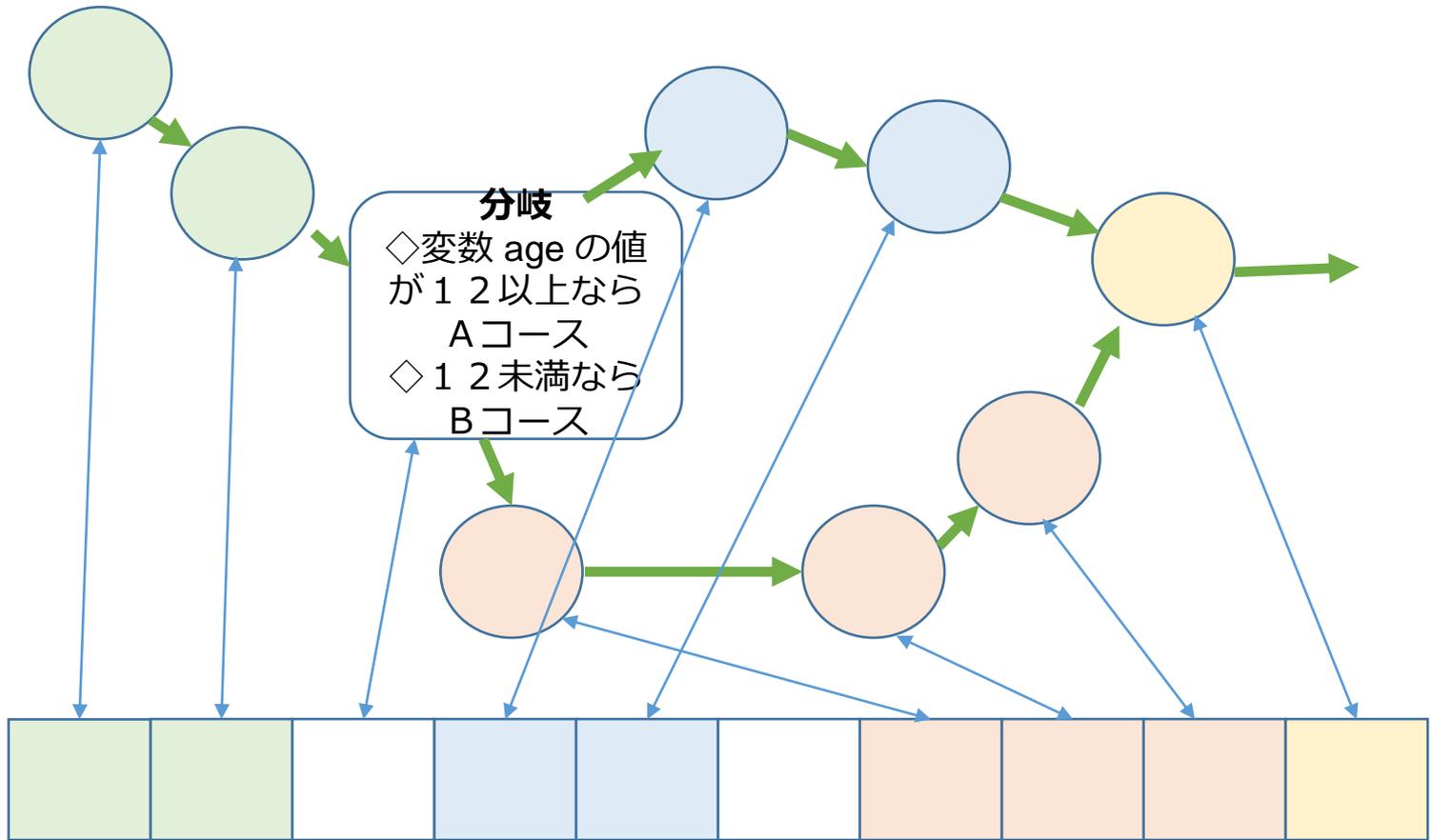
「ある条件」が**成り立てば**Aを、  
成り立たなければBを実行

# プログラム実行の流れ

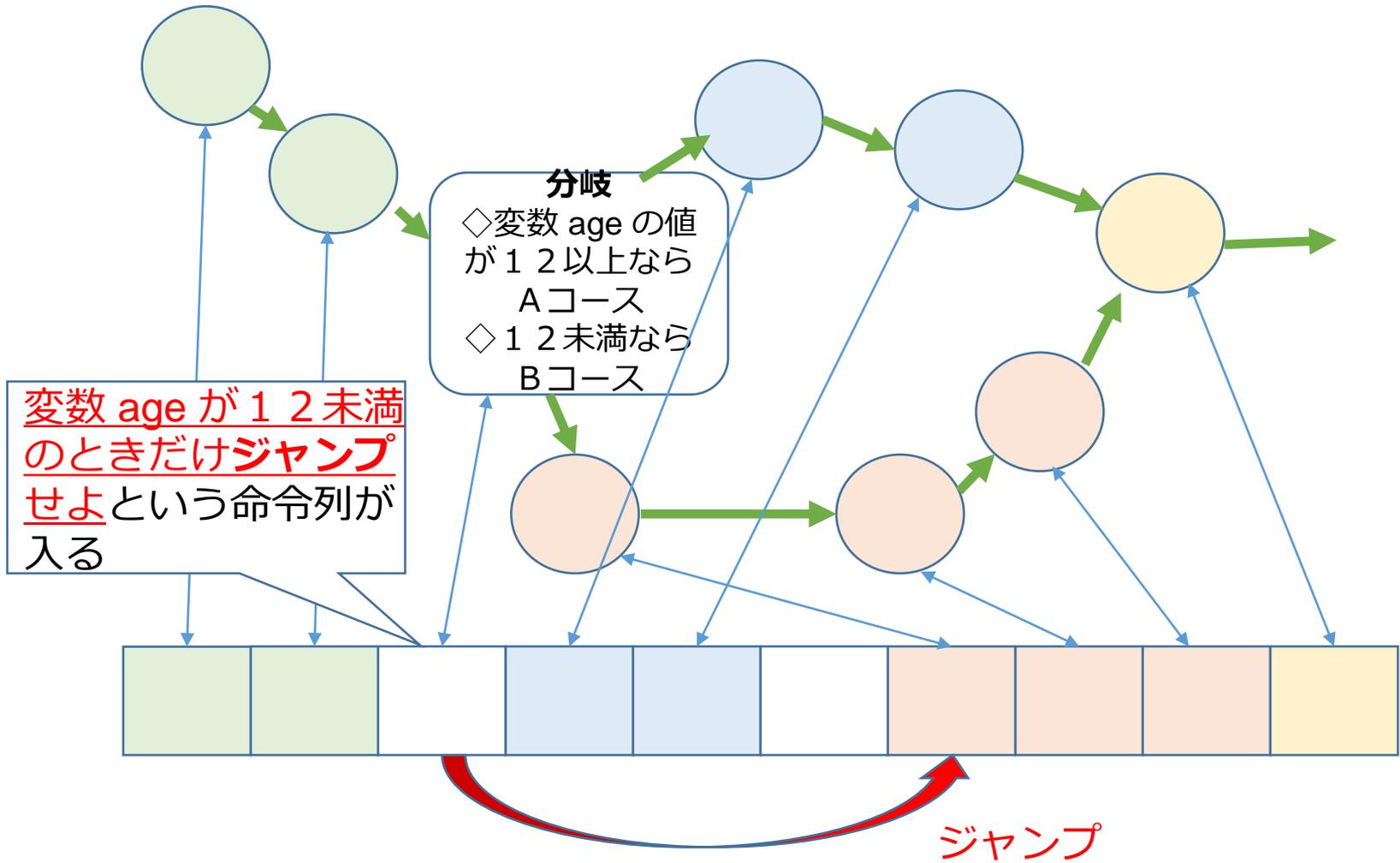


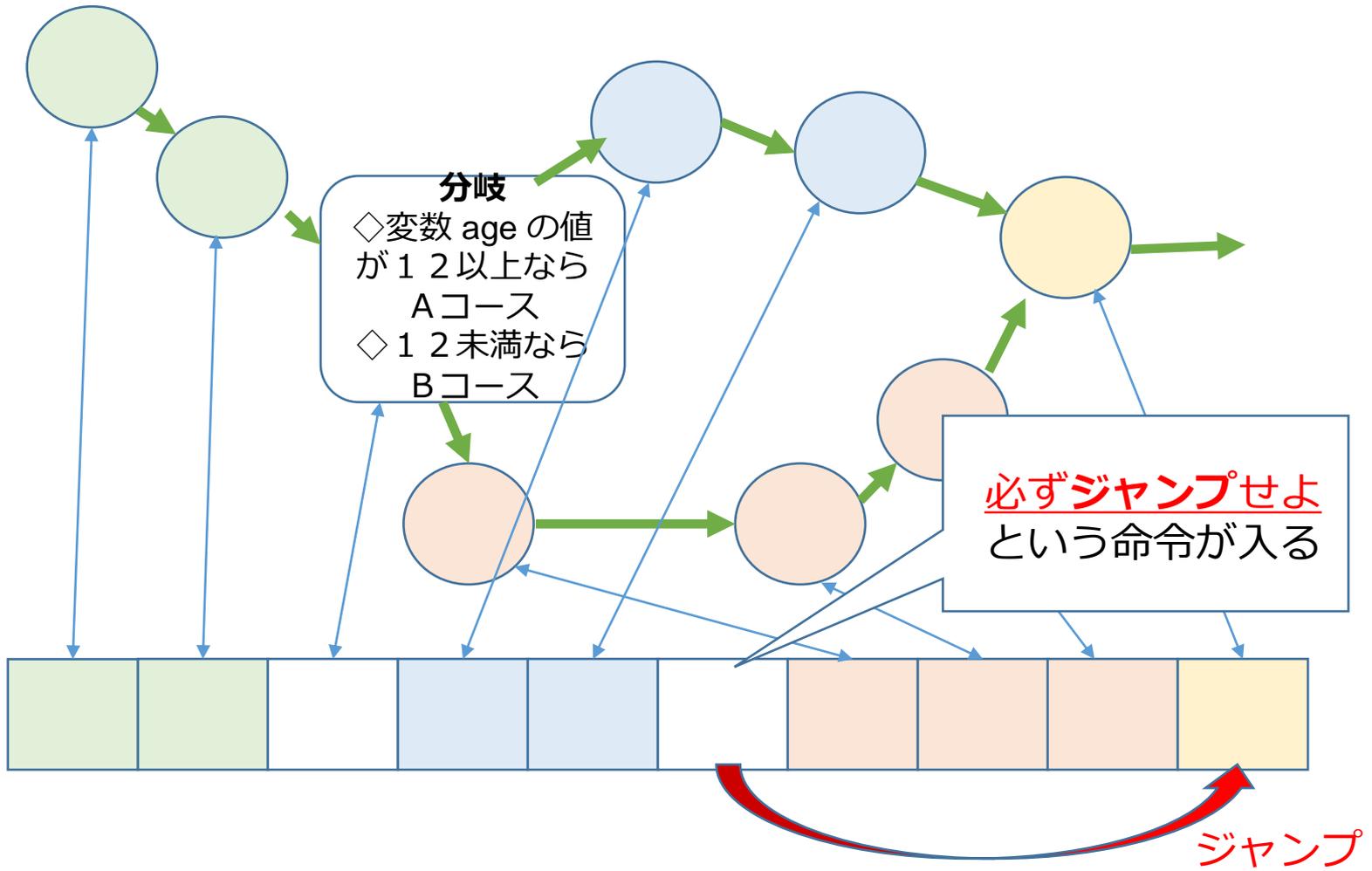
- マシン語（機械語）のプログラム実行は，1度に1命令ずつ進む
- マシン語（機械語）での分岐は，ジャンプ命令の組み合わせ



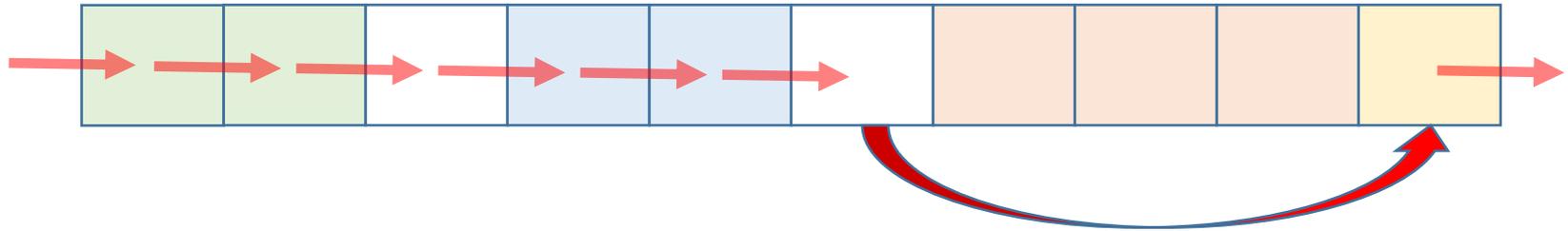


プログラムはメモリに、格納される

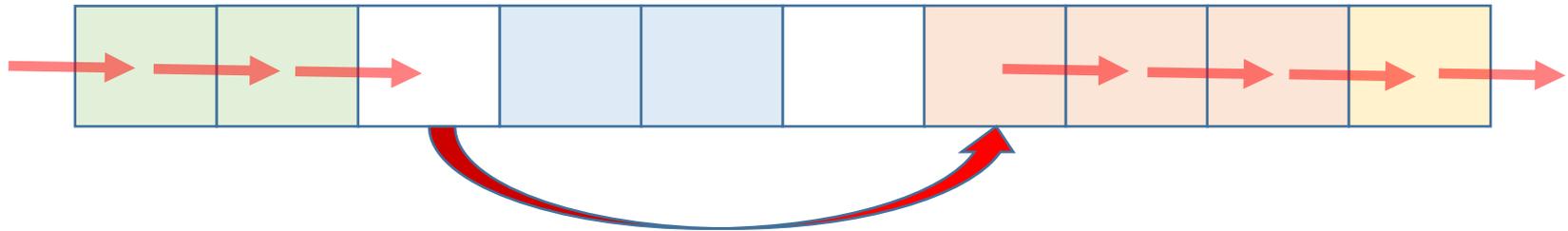




age の値が 12 以上のとき



age の値が 12 未満のとき



# ジャンプ命令の種類



- **無条件ジャンプ命令**

必ずジャンプする

- **条件ジャンプ命令**

**比較命令**の結果によって、ジャンプしたりしなかったりする

# 11-3 条件分岐の演習

## 条件分岐の例

- 12歳以上は 1800円
- 12歳未満は 500円

# Visual C++ のソースファイル例



```
int main()  
{  
    static int age, p;  
    age = 20;  
    if (age > 12)  
        p = 1800;  
    else  
        p = 500;  
    return 0;  
}
```

# Visual C++ 言語とアセンブリ言語



## Visual C++ の プログラム

```
age = 20;
if (age >= 12)
    p = 1800;
else
    p = 500;
```

## アセンブリ言語

同じ意味

同じ意味

同じ意味

```
mov     dword ptr ds:[1258130h],14h
cmp     dword ptr ds:[1258130h],0Ch
jl     wmain+3Dh (012513BDh)
mov     dword ptr ds:[1258134h],708h
jmp     wmain+47h (012513C7h)
mov     dword ptr ds:[1258134h],1F4h
```

命令    命令が対象とする相手である  
オペランド

この2行で「変数 age が 12 未満のときだけジャンプせよ」  
という意味

```
age = 20;  
if (age >= 12)  
    p = 1800;  
else  
    p = 500;
```

```
mov     dword ptr ds:[1258130h],14h  
cmp     dword ptr ds:[1258130h],0Ch  
jl      wmain+3Dh (012513BDh)  
mov     dword ptr ds:[1258134h],708h  
jmp     wmain+47h (012513C7h)  
mov     dword ptr ds:[1258134h],1F4h
```

条件ジャンプ

必ずジャンプせよ  
という無条件ジャンプ命令

```
age = 20;  
if (age >= 12)  
    p = 1800;  
else  
    p = 500;
```

```
mov     dword ptr ds:[1258130h],14h  
cmp     dword ptr ds:[1258130h],0Ch  
jl      wmain+47h (012513BDh)  
mov     dword ptr ds:[1258134h],708h  
jmp     wmain+47h (012513C7h)  
mov     dword ptr ds:[1258134h],1F4h
```

無条件ジャンプ

- 条件分岐でのプログラム実行の流れ（実行順）を確認

ステップオーバー機能を利用

### 実行順

```
age = 20;  
if (age >= 12)  
    p = 1800;  
else  
    p = 500;
```

```
① 0101166E mov     dword ptr [age (01019138h)],14h  
    > +12)  
② 01011678 cmp     dword ptr [age (01019138h)],0Ch  
③ 0101167F jle     main+3Dh (0101168Dh)  
    1800;  
④ 01011681 mov     dword ptr [p (0101913Ch)],708h  
    else  
⑤ 01011688 jmp     main+47h (01011697h)  
    500;  
0101168D mov     dword ptr [p (0101913Ch)],1F4h  
    return 0;  
01011697 xor     eax,eax
```

無条件ジャンプ

この行は飛ばされることを確認

• age = 20;

age = 10;

age の値が変わるとジャンプの様子が変化する

```
① 0101166E mov     dword ptr [age (01019138h)],14h
② 01011678 cmp     dword ptr [age (01019138h)],0Ch
③ 0101167F jle     main+3Dh (0101168Dh)
④ 01011681 mov     dword ptr [p (0101913Ch)],708h
⑤ 0101168B jmp     main+47h (01011697h)
0101168D mov     dword ptr [p (0101913Ch)],1F4h
xor     eax,eax
```

ジャンプ

⑤でジャンプ

```
① 0101166E mov     dword ptr [age (01019138h)],14h
② 01011678 cmp     dword ptr [age (01019138h)],0Ch
③ 0101167F jle     main+3Dh (0101168Dh)
④ 01011681 mov     dword ptr [p (0101913Ch)],708h
⑤ 0101168B jmp     main+47h (01011697h)
0101168D mov     dword ptr [p (0101913Ch)],1F4h
xor     eax,eax
```

ジャンプ

③でジャンプ

- ① Visual Studio を起動しなさい
- ② Visual Studio 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

③ 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()
```

```
{
```

```
    static int age, p;
```

```
    age = 20;
```

```
    if (age > 12)
```

```
        p = 1800;
```

```
    else
```

```
        p = 500;
```

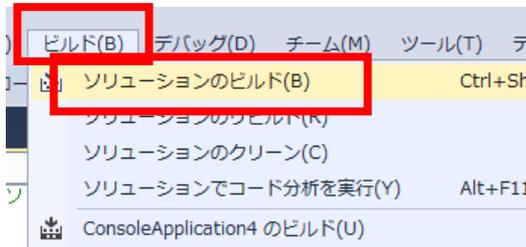
```
    return 0;
```

```
}
```

追加

④ 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



```
出力
出力元(S): ビルド
1>----- ビルド開始: プロジェクト:ConsoleApplication6, 構成:Debug Win32 -----
1> stdafx.cpp
1> ConsoleApplication6.cpp
1> ConsoleApplication6.vcxproj -> e:\documents\visual studio 2015\Projects\
1> ConsoleApplication6.vcxproj -> e:\documents\visual studio 2015\Projects\
===== ビルド: 1 正常終了、0 失敗、0 更新不要、0 スキップ =====
```

## ⑤ Visual Studioで「age = 20;」の行に、ブレークポイントを設定しなさい

```
4 #include "stdatx.h"
5
6
7 int main()
8 {
9     static int age, p;
10    age = 20;
11    if (age > +12)
12        p = 1800;
13    else
14        p = 500;
15    return 0;
16 }
17
```



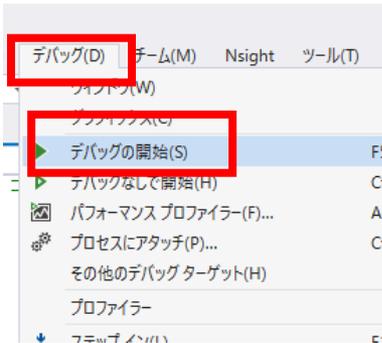
```
7
8
9 int main()
10 {
11     static int age, p;
12     age = 20;
13     if (age > +12)
14         p = 1800;
15     else
16         p = 500;
17     return 0;
18 }
```

① 「age = 20;」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

⑥ 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」

⑦ 「age = 20;」 の行で、実行が中断することを確認しなさい

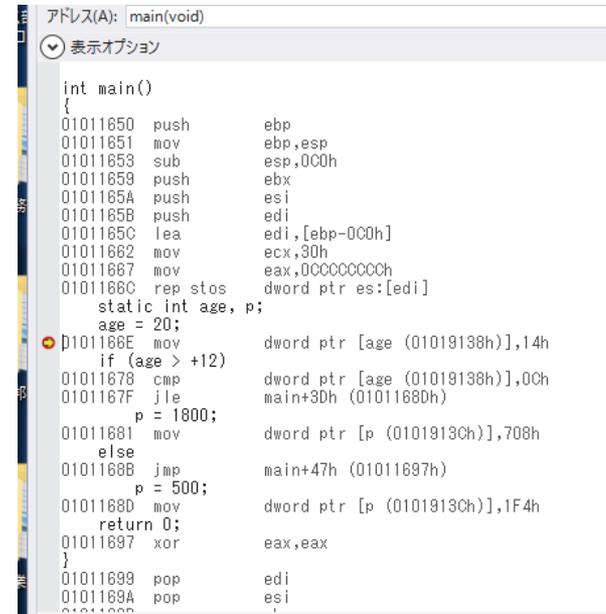
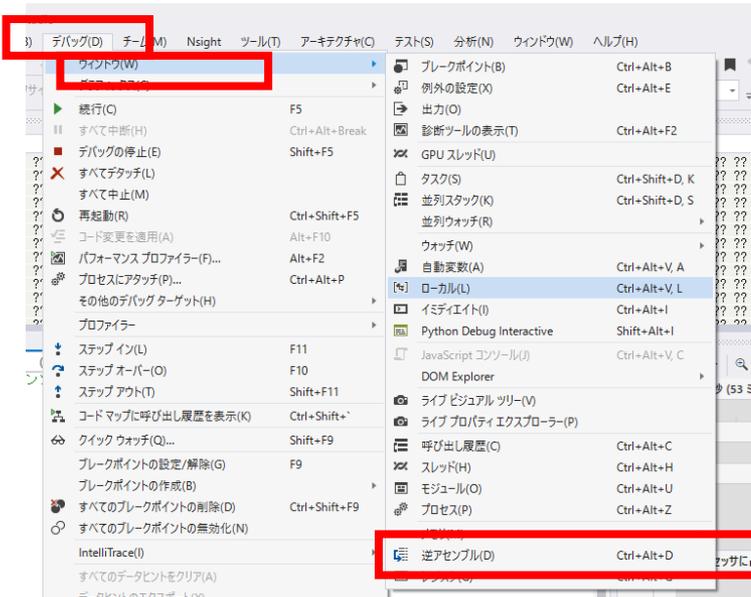
あとで使うので、中断したままにしておくこと



```
int main()
{
    static int age, p;
    age = 20;
    if (age > +12)
        p = 1800;
    else
        p = 500;
    return 0;
}
```

「age = 20;」 の行で実行が  
中断している

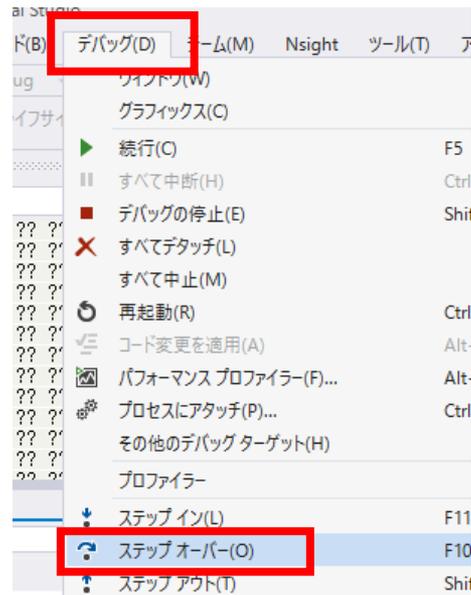
# ⑧ 「age = 20;」 の行で，実行が中断した状態で，逆アセンブルを行いなさい。



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

⑨ ステップオーバーの操作を1回ずつ行いながら、実行の流れを確認しなさい。



「デバッグ」  
→ 「ステップオーバー」  
(あるいは F10 キー)

```

0101166E mov     dword ptr [age (01019138h)],14h
        if (age > +12)
01011678 cmp     dword ptr [age (01019138h)],0Ch
0101167F jle     main+3Dh (0101168Dh)
        p = 1800;
01011681 mov     dword ptr [p (0101913Ch)],708h
        else
0101168B jmp     main+47h (01011697h)
        p = 500;
0101168D mov     dword ptr [p (0101913Ch)],1F4h
        return 0;
01011697 .....

```

↓ ステップオーバー

```

0101166E mov     dword ptr [age (01019138h)],14h
        if (age > +12)
01011678 cmp     dword ptr [age (01019138h)],0Ch
0101167F jle     main+3Dh (0101168Dh)
        p = 1800;
01011681 mov     dword ptr [p (0101913Ch)],708h
        else
0101168B jmp     main+47h (01011697h)
        p = 500;
0101168D mov     dword ptr [p (0101913Ch)],1F4h
        return 0;
01011697 .....

```

↓ ステップオーバー

```

0101166E mov     dword ptr [age (01019138h)],14h
        if (age > +12)
01011678 cmp     dword ptr [age (01019138h)],0Ch
0101167F jle     main+3Dh (0101168Dh) ≤1ミリ秒経過
        p = 1800;
01011681 mov     dword ptr [p (0101913Ch)],708h
        else
0101168B jmp     main+47h (01011697h)
        p = 500;
0101168D mov     dword ptr [p (0101913Ch)],1F4h
        return 0;
01011697 .....

```

↓ ステップオーバー

```

0101166E mov     dword ptr [age (01019138h)],14h
        if (age > +12)
01011678 cmp     dword ptr [age (01019138h)],0Ch
0101167F jle     main+3Dh (0101168Dh)
        p = 1800;
01011681 mov     dword ptr [p (0101913Ch)],708h
        else
0101168B jmp     main+47h (01011697h)
        p = 500;
0101168D mov     dword ptr [p (0101913Ch)],1F4h
        return 0;
01011697 .....

```

↓ ステップオーバー

```

0101166E mov     dword ptr [age (01019138h)],14h
        if (age > +12)
01011678 cmp     dword ptr [age (01019138h)],0Ch
0101167F jle     main+3Dh (0101168Dh)
        p = 1800;
01011681 mov     dword ptr [p (0101913Ch)],708h
        else
0101168B jmp     main+47h (01011697h) ≤1ミリ秒経過
        p = 500;
0101168D mov     dword ptr [p (0101913Ch)],1F4h
        return 0;
01011697 .....

```

↓ ステップオーバー

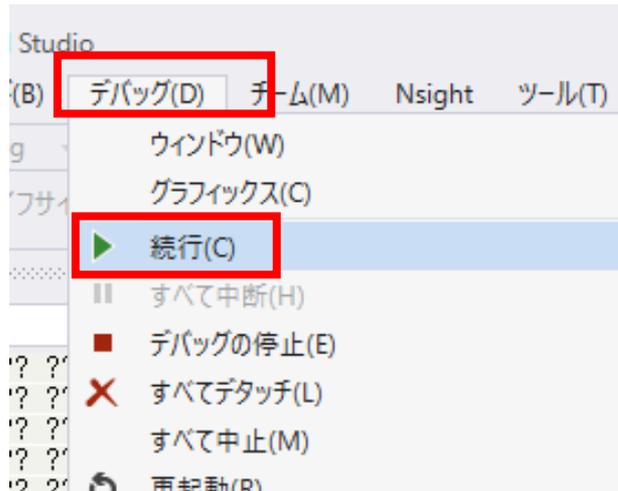
```

0101166E mov     dword ptr [age (01019138h)],14h
        if (age > +12)
01011678 cmp     dword ptr [age (01019138h)],0Ch
0101167F jle     main+3Dh (0101168Dh)
        p = 1800;
01011681 mov     dword ptr [p (0101913Ch)],708h
        else
0101168B jmp     main+47h (01011697h)
        p = 500;
0101168D mov     dword ptr [p (0101913Ch)],1F4h
        return 0;
01011697 xor     eax,eax ≤1ミリ秒経過

```

「mov ...」の行は  
スキップされること  
を確認

⑩ 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」

# 演習課題



「age = 20;」の行を「age = 10;」に変えて、  
今の手順を繰り返さない。

ジャンプの様子が変わるので確認しない

# 11-4 繰り返しの演習

# 繰り返し

- 同じ処理を繰り返し, いつかは終わる

$$y[i] = x[i] * 12$$

繰り返す処理

y	x
36	3
60	5
24	2
12	1
24	2
24	2

i = 0, 1, 2, 3, 4, 5 と変化し  
全部済んだら終わる

終了条件

# 繰り返しの例



## Visual C++ のプログラム

y	x
36	3
60	5
24	2
12	1
24	2
24	2

```
static int x[6] = { 3, 5, 2, 1, 2, 2 };
static int y[6];
int i;
for (i = 0; i < 6; i++) {
    y[i] = x[i] * 12;
}
```

繰り返す処理

i の値は  $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$   
と変化し, 全部済んだら終わる

# Visual C++ 言語とアセンブリ言語



## Visual C++ の プログラム

```
int _tmain(int argc, _TCHAR* argv[])
{
    static int x[6] = { 3, 5, 2, 1, 2, 2 };
    static int y[6];
    int i;
    for (i = 0; i < 6; i++) {
        y[i] = x[i] * 12;
    }

    return 0;
}
```

同じ意味

## アセンブリ言語

```
00C7166E mov     dword ptr [i],0
00C71675 jmp     main+30h (0C71680h)
00C71677 mov     eax,dword ptr [i]
00C7167A add     eax,1
00C7167D mov     dword ptr [i],eax
00C71680 cmp     dword ptr [i],6
00C71684 jge     main+40h (0C71690h)
...
00C71686 mov     eax,dword ptr [i]
00C71689 imul  ecx,dword ptr x (0C79000h)[eax*4],0Ch
00C71691 mov     edx,dword ptr [i]
00C71694 mov     dword ptr y (0C79158h)[edx*4],ecx
...
00C7169B jmp     main+27h (0C71677h)
}
```

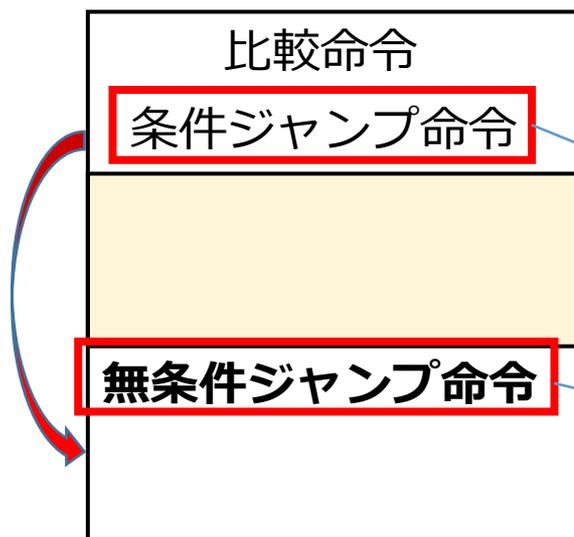
命令

命令が対象とする  
相手であるオペランド

# ジャンプ命令



## アセンブリ言語



```
00C7166E mov     dword ptr [i],0
00C71675 jmp     main+30h (0C71680h)
00C71677 mov     eax,dword ptr [i]
00C7167A add     eax,1
00C7167D mov     dword ptr [i],eax
00C71680 cmp     dword ptr [i],8
00C71684 jge     main+40h (0C71690h)
00C71686 mov     eax,dword ptr [i]
00C71689 imul  ecx,dword ptr x (0C79000h)[eax*4],0Ch
00C71691 mov     edx,dword ptr [i]
00C71694 mov     dword ptr y (0C79158h)[edx*4],ecx
00C7169B jmp     main+27h (0C71677h)
```

命令

命令が対象とする  
相手であるオペランド

- 条件分岐でのプログラム実行の流れ（実行順）を確認

ステップオーバー機能を利用

- ① Visual Studio を起動しなさい
- ② Visual Studio 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

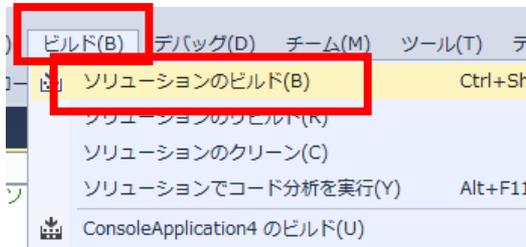
### ③ Visual Studioのエディタを使って、ソースファイルを編集しなさい

```
int main()  
{  
    static int x[6] = { 3, 5, 2, 1, 2, 2, };  
    static int y[6];  
    int i;  
    for (i = 0; i < 6; i++) {  
        y[i] = x[i] * 12;  
    }  
}
```

追加

④ 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



```
出力
出力元(S): ビルド
1>----- ビルド開始: プロジェクト:ConsoleApplication6, 構成:Debug Win32 -----
1> stdafx.cpp
1> ConsoleApplication6.cpp
1> ConsoleApplication6.vcxproj -> e:\documents\visual studio 2015\Projects\
1> ConsoleApplication6.vcxproj -> e:\documents\visual studio 2015\Projects\
===== ビルド: 1 正常終了、0 失敗、0 更新不要、0 スキップ =====
```

## ⑤ Visual Studioで「for」の行に、ブレークポイントを を設定しなさい



The image shows three parts of the Visual Studio interface:

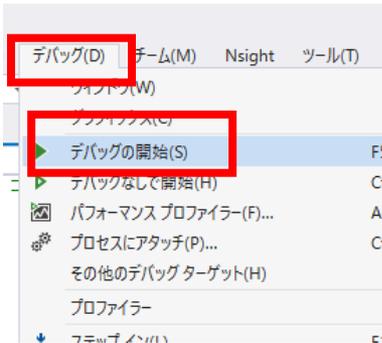
- Code Editor (Left):** A C# code snippet with a `for` loop. The line `for (i = 0; i < 8; i++) {` is highlighted with a red box.
- Debug Menu (Middle):** The 'デバッグ(D)' menu is open, and the option 'ブレークポイントの設定/解除(G)' is highlighted with a red box.
- Code Editor (Right):** The same code snippet is shown, but with a red dot (breakpoint) placed on the line number 12, corresponding to the `for` loop line.

① 「for (i = 0;」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

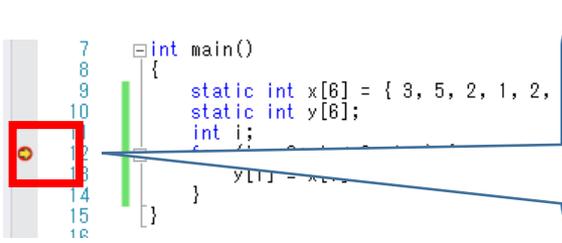
⑥ Visual Studioで、デバッガーを起動しなさい。



「デバッグ」  
→ 「デバッグ開始」

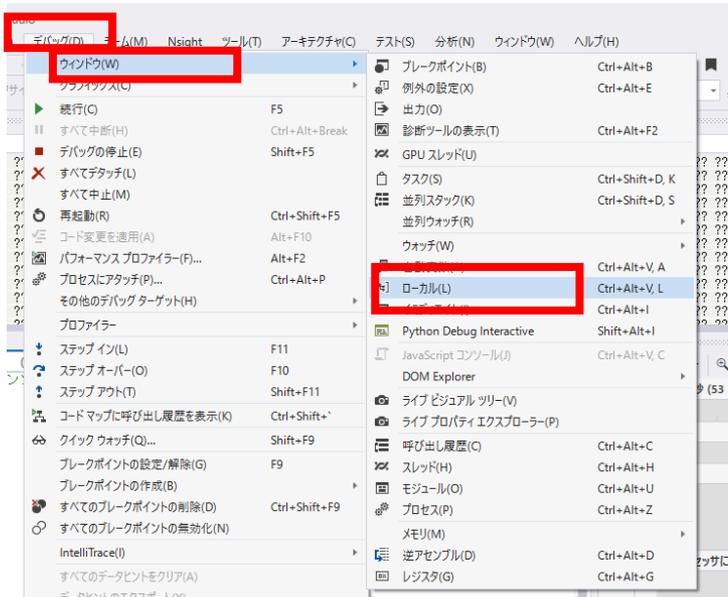
⑦ 「for」の行で、実行が中断することを確認しなさい

あとで使うので、中断したままにしておくこと



「for (i = 0; ...」の行で実行が中断している

⑧ 「for」 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．



ローカル	
名前	値
i	-858993460
x	0x00c79000 {3, 5, 2, 1, 2, 2}
y	0x00c79158 {0, 0, 0, 0, 0}

② 変数名と値の対応表が表示される

① 「デバッグ」  
→ 「ウインドウ」  
→ 「ローカル」

※ 次ページに拡大図

## ローカル

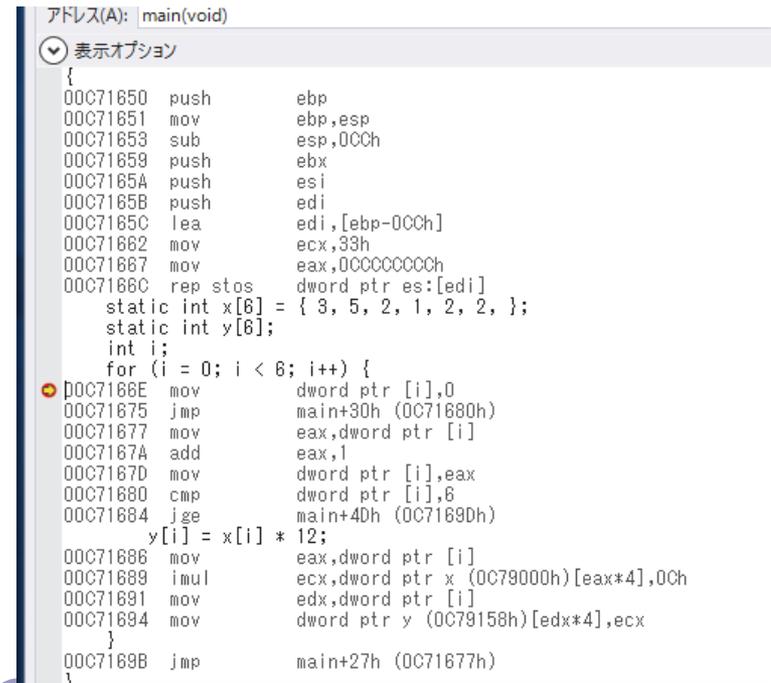
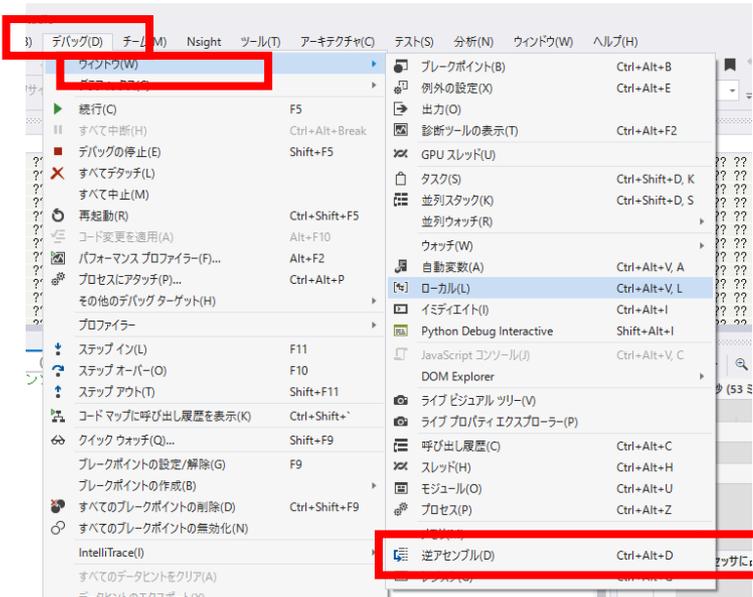
名前

変数  $i$  の値は、変な値になっ  
ているはず

	$i$	-858993460
▷ 	$x$	0x00c79000 {3, 5, 2, 1, 2, 2}
▷ 	$y$	0x00c79158 {0, 0, 0, 0, 0, 0}

「 $y[i] = x[i] * 12;$ 」は  
未実行であることを確認！

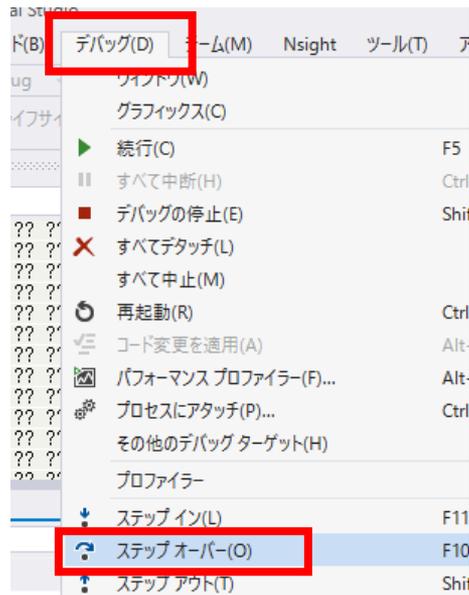
# ⑨ 「for」 の行で，実行が中断した状態で，逆アセンブルを行いなさい。



① 「デバッグ」 → 「ウインドウ」 → 「逆アセンブル」

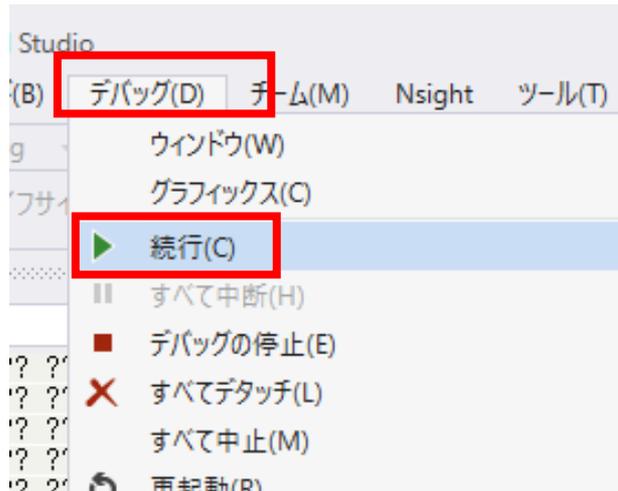
② 逆アセンブルの結果が表示される

⑩ ステップオーバーの操作を1回ずつ行いながら、実行の流れを確認しなさい。



「デバッグ」  
→ 「ステップオーバー」  
(あるいは F10 キー)

⑪ 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」

# 確認クイズ



- ジャンプ命令は、2つ確認できましたか
- ジャンプ命令でジャンプが行われたことを確認できましたか

# 演習

# 足し算の繰り返し



- 次のプログラムで、プログラム実行の流れを Visual Studio で確認しなさい

```
static int a[4] = {1, 2, 3, 4};  
static int b[4] = {2, 4, 6, 8};  
static int c[4];  
int i;  
for (i = 0; i < 4; i++) {  
    c[i] = a[i] + b[i];  
}
```

足し算を 4 回繰り返すプログラム。  
結果は  
3, 6, 9, 12

# 足し算の繰り返し



- 次のプログラムで, プログラム実行の流れを Visual Studio で確認しなさい

```
static int a[4] = {1, 2, 3, 4};
```

```
int s;
```

```
int i;
```

```
s = 0;
```

```
i = 0;
```

```
while (s < 5) {
```

```
    s = s + a[i];
```

```
    i++;
```

```
}
```

1, 2, 3, 4 の合計を求めるプログラム.

結果は

10

# 文字列の長さ



- 文字列の長さを数えるプログラム

```
static char s[10] = "abc";  
int i;  
for (i=0;;i++)  
    if ( s[i] == 0 )  
        break;  
printf("%d¥n", i);
```