ca-10. フラグ，フラグレジスタ

（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

10-1 フラグ

フラグの用途の例

自動で分岐させたい

→　**フラグ**を使う

フラグ

フラグを立てる
(セット)

フラグを下げる
(クリア)

フラグ

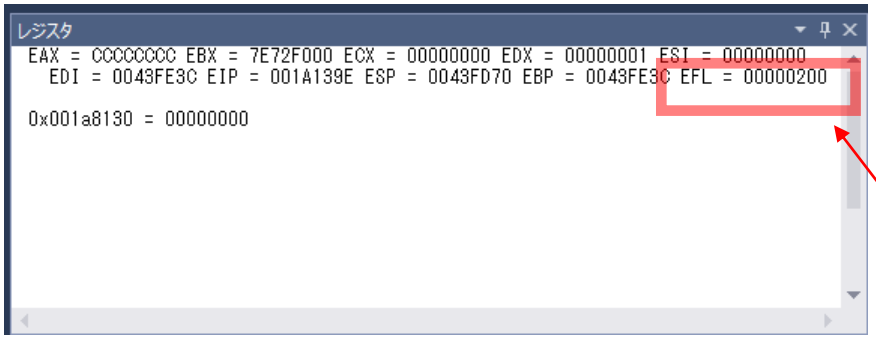
- フラグの値は　0　　または　　1

- プログラムの進行を決めるのに利用できる

- フラグの値は，自動で変化する

10-2 フラグレジスタ

フラグレジスタ



フラグレジスタは，
レジスタウインドウ「EFL」

この値が変化

フラグレジスタ （名前は EFLAGS で、縮めて EFL）に、
たくさんのフラグが入っている

フラグの名前	ID	VIP	VIF	AC	VM	RF		NT	IOPL		OF	DF	IF	TF	SF	ZF		AF		PF		CF
値	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

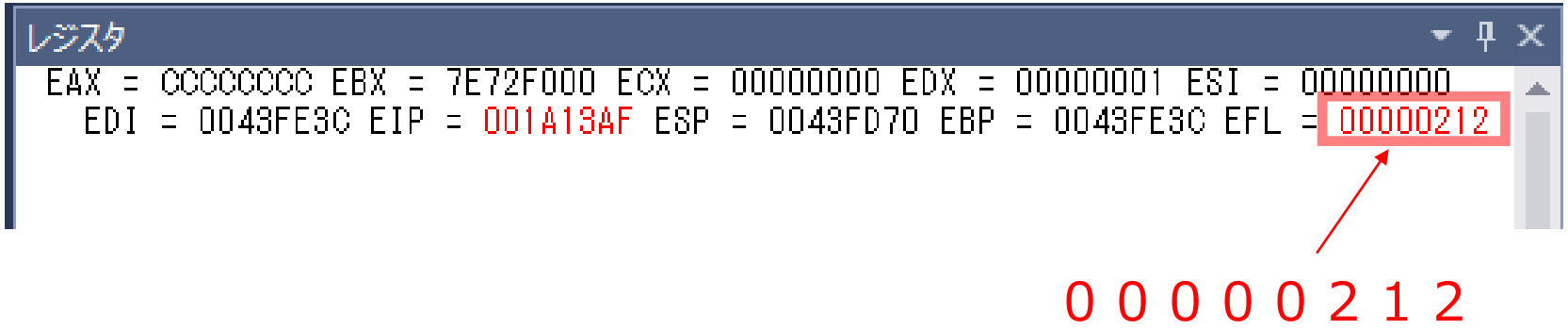
長さ 32 ビット

フラグレジスタ

- 桁上がりフラグ (Carry Flag, **CF**)
  - 演算結果に桁上がりが生じると 1
- 桁あふれフラグ (Overflow Flag, **OF**)
  - 演算結果に桁あふれが生じると 1
- ゼロフラグ (Zero Flag, **ZF**)
  - 演算結果がゼロになると 1
- サインフラグ (Sign Flag, **SF**)
  - 演算結果がマイナスの数になると 1

など

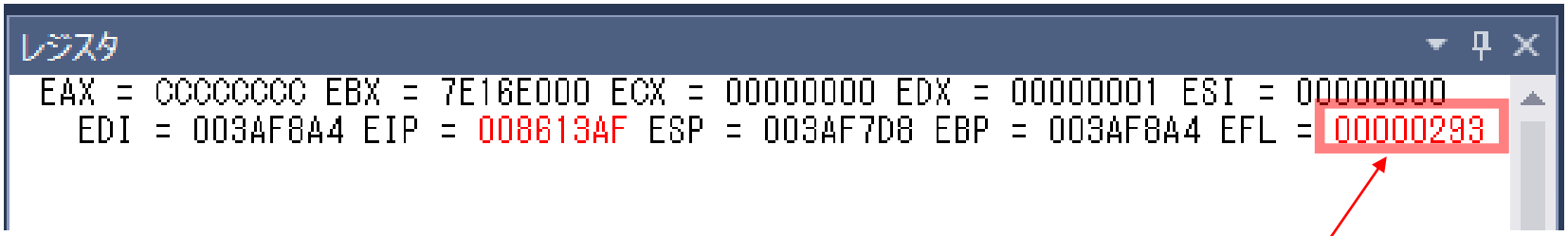
フラグレジスタ



0 0 0 0 0 2 1 2

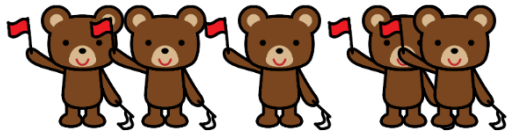
フラグの名前	ID	VIP	VIF	AC	VM	RF		NT	IOPL	OF	DF	IF	TF	SF	ZF		AF		PF		CF
値	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0

フラグレジスタ



0 0 0 0 0 2 9 3

フラグの名前	ID	VIP	VIF	AC	VM	RF		NT	IOPL		OF	DF	IF	TF	SF	ZF		AF		PF		CF
値	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1	1



フラグレジスタ

- フラグレジスタのビット１つ１つが，フラグになっている

10-3 Visual Studio で フラグの変化を見る

今から行うこと

- フラグの値（0 または1）の変化を見る

- フラグの値がある条件のときだけジャンプする命令（条件ジャンプ命令）を見る

簡単な条件分岐の例

- 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;
}
```

パソコン演習

では，フラグレジスタ (EFL) の値の変化と，ジャンプの様子を確認してください

ステップオーバー機能を利用

演習

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

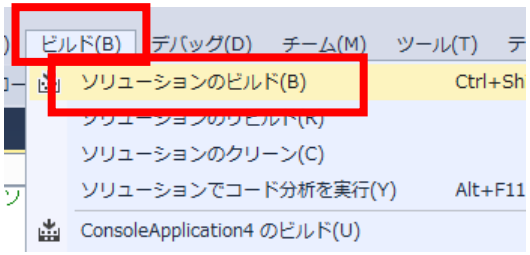
③ Visual Studioのエディタを使って，ソースファイルを編集しなさい

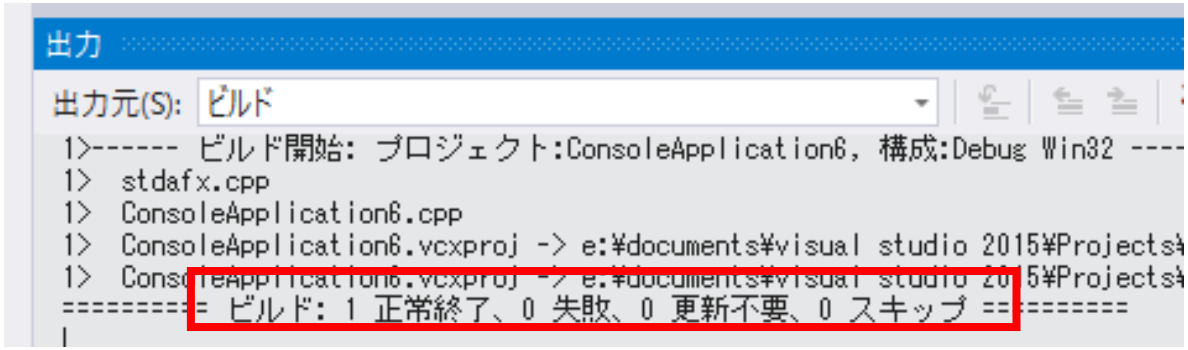
```
int main()
{
    static int age, p;
    age = 20;
    if (age > 12)
        p = 1800;
    else
        p = 500;
    return 0;
}
```

追加

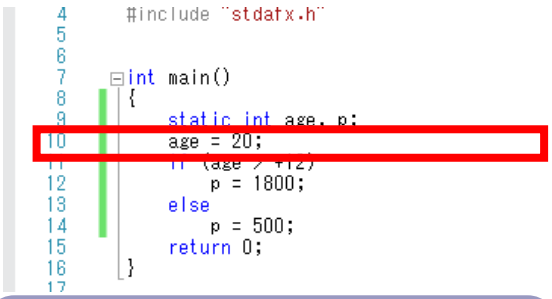
④ ビルドしなさい．ビルドのあと「１　正常終了，０　失敗」の表示を確認しなさい

→　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



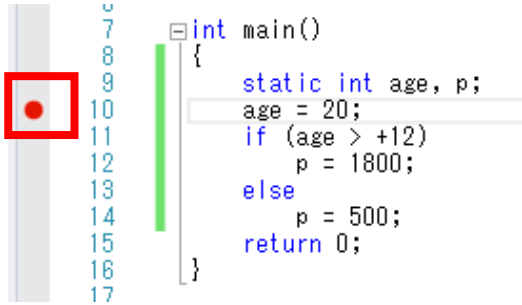


⑤ Visual Studioで「age = 20;」の行に，ブレークポイントを設定しなさい



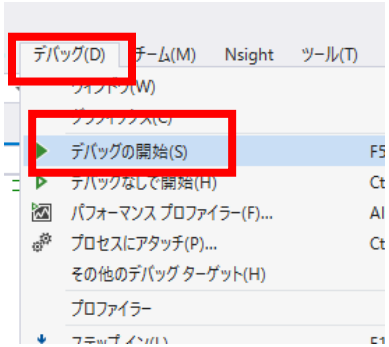
① 「**age = 20;**」の行をマウスでクリック

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

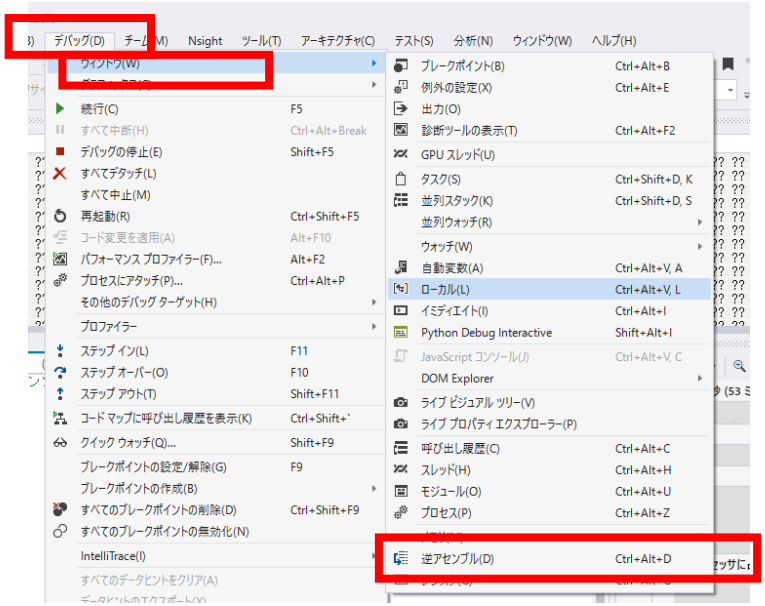
あとで使うので，中断したままにしておくこと

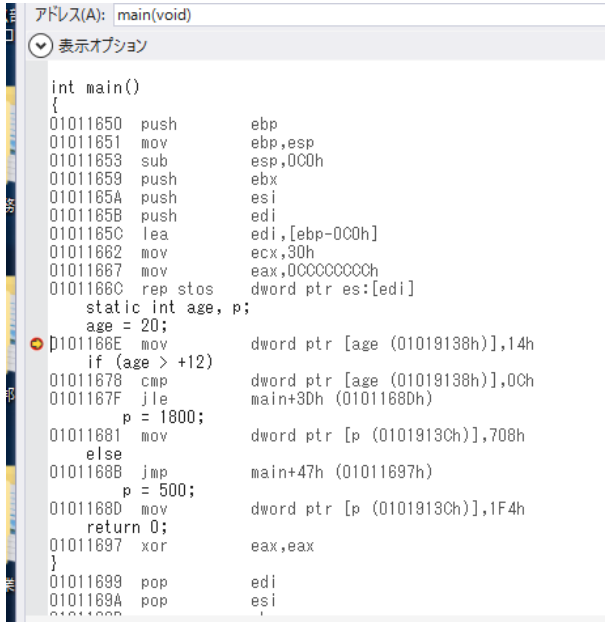
黄色の矢印

```
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
```

「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.





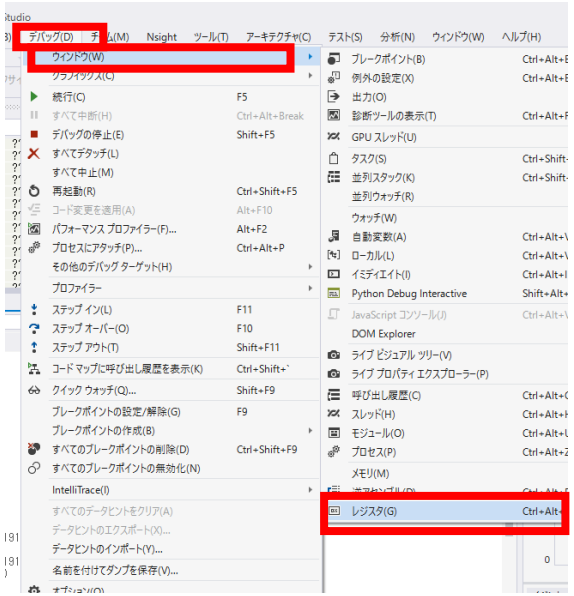
① 「デバッグ」→「ウインドウ」→「**逆アセンブル**」

② 逆アセンブルの結果が表示される

⑨「age = 20;」の行で，実行が中断した状態で，レジスタの中身を表示させなさい．手順は次の通り．

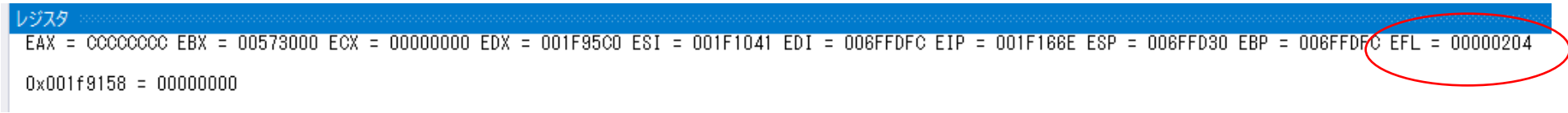
```
01011678  cmp         dword ptr [age (01019138h)],0Ch
0101167F  jle         main+3Dh (0101168Dh)
        p = 1800;
01011681  mov         dword ptr [p (0101913Ch)],708h
    else
01011688  jmp         main+47h (01011697h)
        p = 500;
0101168D  mov         dword ptr [p (0101913Ch)],1F4h
    return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

デバッガーを**起動済み**で，プログラムの実行が中断しているときに・・・



① 「デバッグ」
→「ウインドウ」→「**レジスタ**」

② レジスタが表示される．
EFL に注目



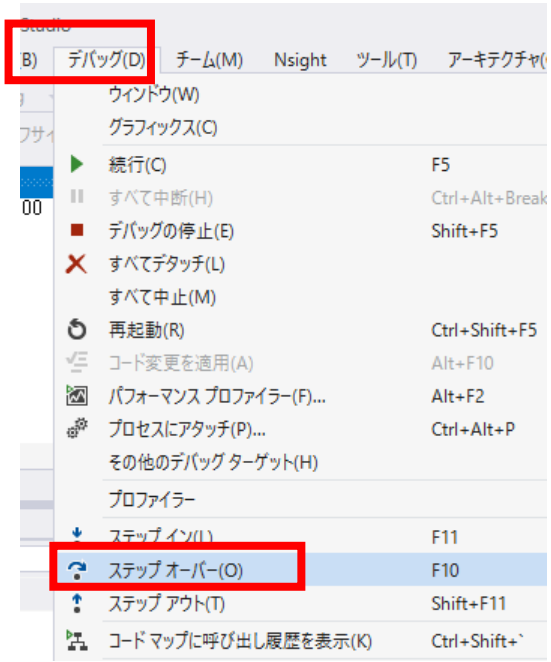
EFL が**フラグレジスタ**

⑩ステップオーバーの操作を１回ずつ行いながら，

・プログラムカウンタ（黄色の矢印）

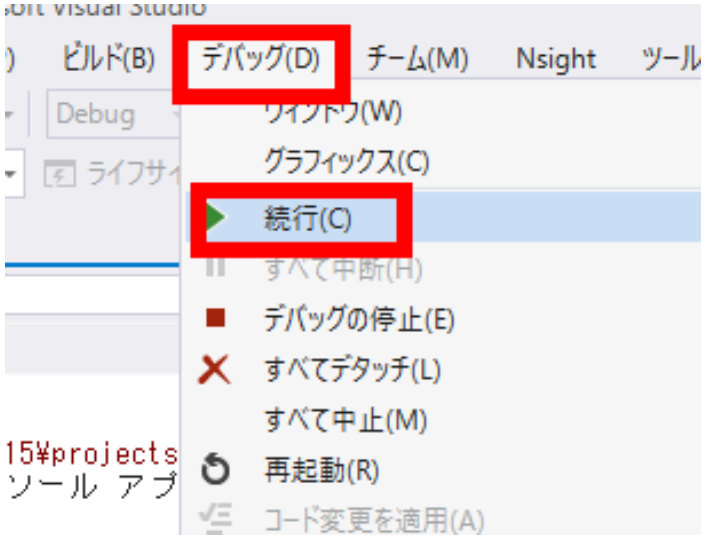
・フラフレジスタ（レジスタウインドウの中のEFL）

の変化を確認しなさい.



「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**」

⑫ 今度は，「age = 20;」の行を「age = 10;」に変えて.

ステップオーバーの操作を 1 回ずつ行いながら，

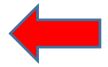
・プログラムカウンタ（黄色の矢印）

・フラフレジスタ（レジスタウインドウの中の EFL）

の変化を確認しなさい.

Visual C++ の
プログラム

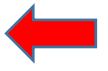
```
age = 20;
if (age >= 12)
    p = 1800;
else
    p = 500;
```

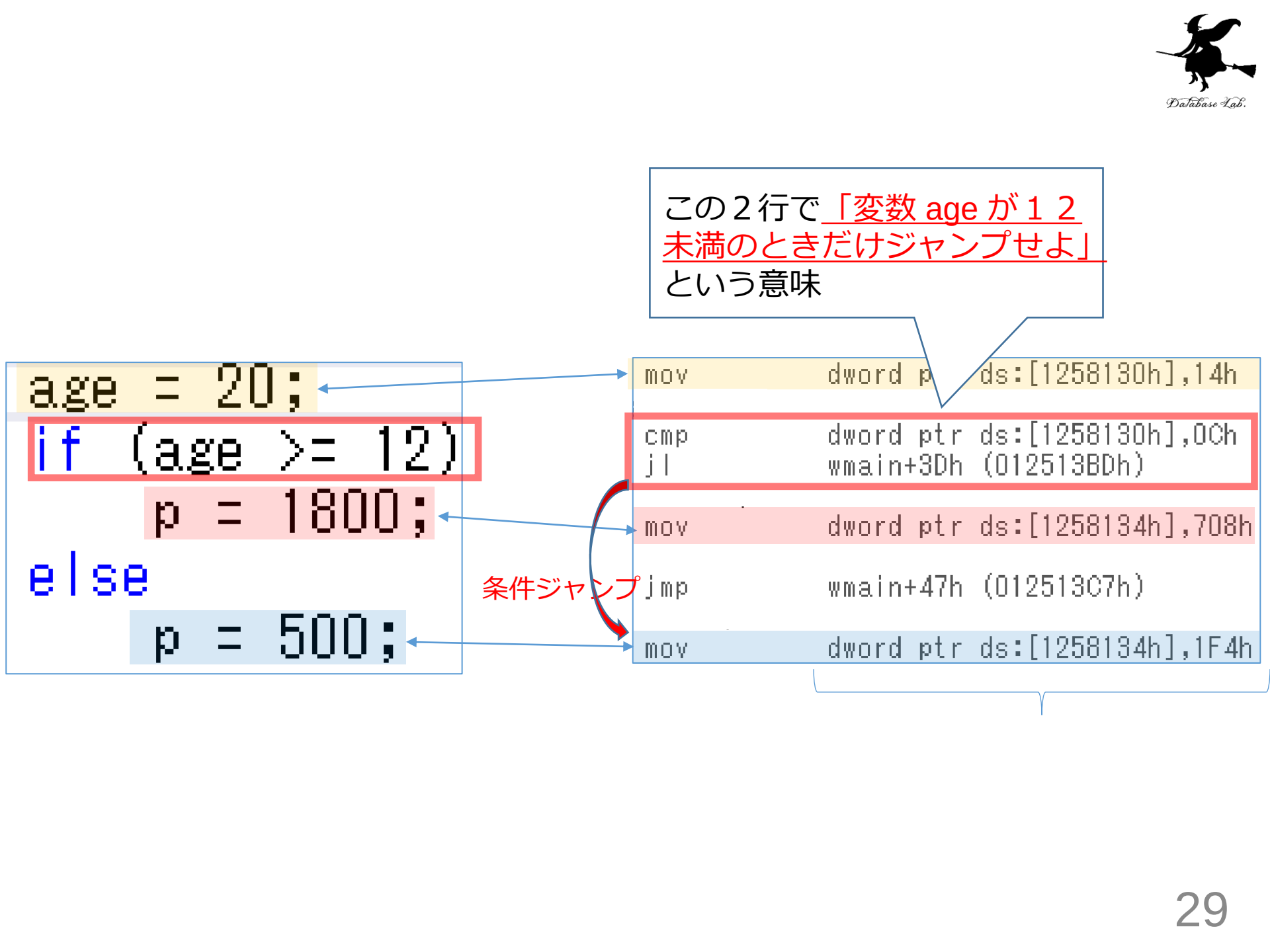

こちらが
有効

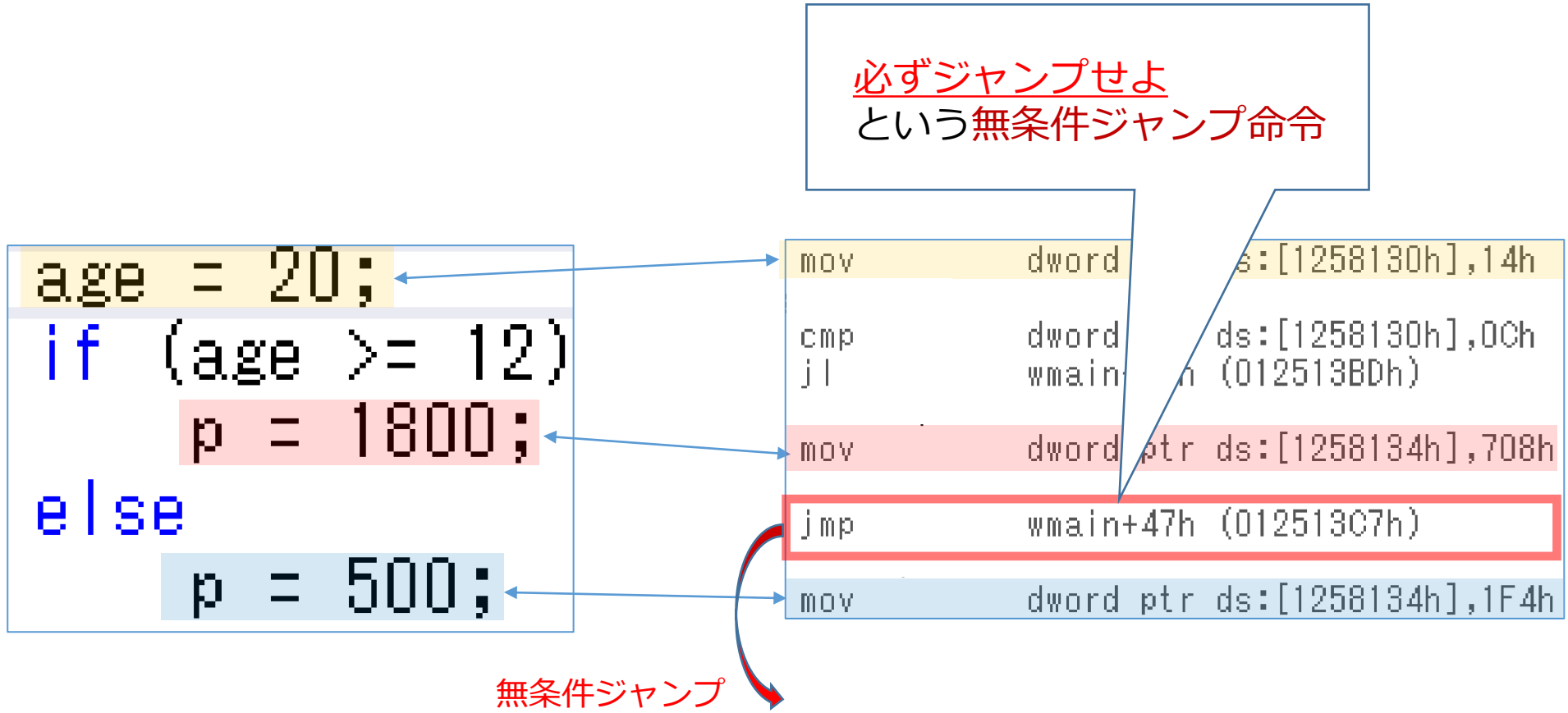

無視
される

```
age = 10;
if (age >= 12)
    p = 1800;
else
    p = 500;
```


無視
される


こちらが
有効


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;
条件ジャンプ
mov dword p 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


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 s:[1258130h],14h
cmp dword ds:[1258130h],0Ch
jl wmain h (012513BDh)
mov dword ptr ds:[1258134h],708h
jmp wmain+47h (012513C7h)
mov dword ptr ds:[1258134h],1F4h
無条件ジャンプ