

# 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



















#### フラグを立てる (セット)

#### フラグを下げる (クリア)







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

#### 長さ 32 ビット

# フラグレジスタ

Database Lab.

- 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





・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<u>以上</u>は1800円
- ・12歳<u>未満</u>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 O;
```





# 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





### 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

	ال	∠ド(B)	デバッグ(D)	チーム(M)	ツール	(T)	テ
1-	Ē.	-בעע	ションのビル	<sup>к</sup> (В)		Ctrl+	Sh
	-	ンワエー	ションのウビ	VI^(N)			
		-בעע	-ションのクリ-	-ン(C)			
$\overline{\mathcal{Y}}$		ソリューションでコード分析を実行(Y) Alt+F1			F11		
	*	Console	Application4 0	のビルド(U)			

出力	
出力元(S): ビルド	• 🖳 🖆 🛬
<ul> <li>1&gt; ビルド開始: プロジェクト:ConsoleApplication6,</li> <li>1&gt; stdafx.cpp</li> <li>1&gt; ConsoleApplication6.cpp</li> </ul>	構成:Debug Win32
1> ConsoleApplication6.vcxproj -> e:¥documents¥visual 1> ConsoleApplication6.vcxproj -> e.¥documents¥visual ====================================	studio 2015¥Projectsà studio 2015¥Projectsà スキップ == <mark></mark> =======



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





# ⑥ Visual Studioで, デバッガーを起動しなさい.



## 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





## 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.

<ol> <li>デ/・</li> <li>ディング・</li> <l< th=""><th>(マグ(D) チー/ M) Nsight ツール(T) マイントラ(W) ボデン・トゥック 続行(C) すべて中断(H) デバッグの停止(E) すべて中か(H) オ、マキル(M) 再起動(R) コード変更を適用(A) パ(フォーマンスプロファイラー(F) プロセスにアタッチ(P) その他のデバッグターグット(H)</th><th>アーキテグチャ(C)     ・      ・</th><th><ul> <li>テスト(S) 分析(N) ウインドク()</li> <li>         ブレークポイント(B)     </li> <li>         ブレークポイント(B)     </li> <li>         「例外の設定(X)     </li> <li>         単力(O)     </li> <li>         「動デリールの表示(T)     </li> <li>         ズ GPU スレッド(U)     </li> <li>         クスク(S)     </li> <li>         「動列スクック(K)     </li> <li>         並列スクック(K)     </li> <li>         並列スクック(K)     </li> <li>         立利スクック(K)     </li> <li>         直向動変数(A)     </li> <li>         「マナイエイト(I)     </li> </ul></th><th>W)         ∧JJ/J(H)           Ctrl+Alt+B            Ctrl+Alt+F2            Ctrl+Alt+F2            Ctrl+Alt+F2            Ctrl+Alt+F2            Ctrl+Alt+F2            Ctrl+Shift+D, K         ?           Ctrl+Shift+D, S         ?           Ctrl+Alt+V, A         ?           Ctrl+Alt+V, A         ?           Ctrl+Alt+V, L         ?           Ctrl+Alt+V, L         ?           Ctrl+Alt+V, L         ?</th></l<></ol>	(マグ(D) チー/ M) Nsight ツール(T) マイントラ(W) ボデン・トゥック 続行(C) すべて中断(H) デバッグの停止(E) すべて中か(H) オ、マキル(M) 再起動(R) コード変更を適用(A) パ(フォーマンスプロファイラー(F) プロセスにアタッチ(P) その他のデバッグターグット(H)	アーキテグチャ(C)     ・      ・	<ul> <li>テスト(S) 分析(N) ウインドク()</li> <li>         ブレークポイント(B)     </li> <li>         ブレークポイント(B)     </li> <li>         「例外の設定(X)     </li> <li>         単力(O)     </li> <li>         「動デリールの表示(T)     </li> <li>         ズ GPU スレッド(U)     </li> <li>         クスク(S)     </li> <li>         「動列スクック(K)     </li> <li>         並列スクック(K)     </li> <li>         並列スクック(K)     </li> <li>         立利スクック(K)     </li> <li>         直向動変数(A)     </li> <li>         「マナイエイト(I)     </li> </ul>	W)         ∧JJ/J(H)           Ctrl+Alt+B            Ctrl+Alt+F2            Ctrl+Alt+F2            Ctrl+Alt+F2            Ctrl+Alt+F2            Ctrl+Alt+F2            Ctrl+Shift+D, K         ?           Ctrl+Shift+D, S         ?           Ctrl+Alt+V, A         ?           Ctrl+Alt+V, A         ?           Ctrl+Alt+V, L         ?           Ctrl+Alt+V, L         ?           Ctrl+Alt+V, L         ?
	プロファイラー ステップ イン(L) ステップ オー/(-(O) ステップ アウト(T) コードマップに呼び出し履歴を表示(K) クイック ウォッチ(Q) ブレークポイントの設定/解除(G) ブレークポイントの使成(B) すべてのブレークポイントの削除(D) すべてのブレークポイントの削除(D) すべてのブレークポイントの削除(D) すべてのブレークポイントの制除(D) すべてのブレークポイントの制除(D)	F11 F10 Shift+F11 Ctrl+Shift+* Shift+F9 F9 Ctrl+Shift+F9	■         Python Debug Interactive           JavaScript コンソール(/)         DOM Explorer           ●         ライブ どジュアルッリー(/)           ●         ライブ ビジュアルッリー(/)           ●         ライブ ゴンパラィ エクスプローラー           ●         ザび出し厚屈(C)           >>         スレッド(H)           ■         モジュール(O)           ●         プロセス(P)           ・         ビアセンブル(D)	Ctrl+Alt+V, C Ctrl+Alt+V, C Ctrl+Alt+V, C Ctrl+Alt+C Ctrl+Alt+H Ctrl+Alt+H Ctrl+Alt+Z Ctrl+Alt+Z Ctrl+Alt+D 27 JT (T

① 「デバッグ」→ 「ウインド ウ | → 「**逆アセンブル** |

ア	ドレス(A): ma	in(void)	
2	) 表示オプショ	ン	
	int main() {	)	
	01011650 01011651 01011653	push mov sub	ebp ebp,esp esp.000b
	01011659 0101165A	push push	ebx esi
	01011658 0101165C 01011662	push lea mov	edi edi,[ebp-OCOh] ecx.30h
	01011667 0101166C	mov rep stos	eax,OCCCCCCCCh dword ptr es:[edi]
	statio	c int age, p; 20•	;
D	p101166E	mov re > +12)	dword ptr [age (01019138h)],14h
	01011678 0101167F	cmp jle	dword ptr [age (01019138h)],OCh main+3Dh (0101168Dh)
	01011681	= 1800; mov	dword ptr [p (0101913Ch)],708h
	0101168B	jmp = 500:	main+47h (01011697h)
	0101168D return	mov n O;	dword ptr [p (O1O1913Ch)],1F4h
	01011697 }	xor	eax,eax
	01011699 0101169A	qoq qoq	edi esi

#### 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.



dword ptr [age (01019138h)],14h

dword ptr [age (01019138h)],OCh main+3Dh (0101168Dh)

dword ptr [p (0101913Ch)],708h

main+47h (01011697h)

dword ptr [p (0101913Ch)],1F4h

eax,eax

デバッガーを起動済みで,

ているときに・・

プログラムの実行が中断し

11	(77(D) 7 a(M) Naight 7 /2(1		_ /^		. 000 (00)
	ウインドウ(W)			ブレークポイント(B)	Ctrl+Alt-
1	(())))))(())		*	例外の設定(X) 出力(O)	Ctrl+Alt-
	統11(5)	Ctrl : Alt : Brook		エガ(0) 診断に止めまその	Christ Alt
1.	デバッグの停止(5)	Chiff . ES	(LM)	20117-1003cA(1)	CUITAIC
1 🗸	すべてデタッチ(1)	Shirters	~	GPU XUYF(U)	
	すべて由止(M)			970(S)	Ctrl+Shi
5	再起動(R)	Ctrl+Shift+E5	(:=	並列スタック(K)	Ctrl+Shif
√ <u>−</u>	- 「変更を適用(A)	Alt+F10		亚列ウオッナ(K)	
	パフォーマンス プロファイラー(F)	Alt+F2	_	ウォッチ(W)	
	プロセスにアタッチ(P)	Ctrl+Alt+P	1	自動変数(A)	Ctrl+Alt+
2	その他のデバッグターゲット(H)		[**]	□-תועת-L)	Ctrl+Alt-
2	プロファイラー			イミテイエイト(I)	Ctrl+Alt+
•	7======	511	711	Python Debug Interactive	Shift+Al
	ステップオーバー(O)	F10	1	JavaScript コンソール(J)	Ctrl+Alt-
	ステップ アウト(T)	Shift+F11		DOM Explorer	
A	コードマップに呼び出し履歴を表示(K)	Ctrl+Shift+`	0	ライブ ビジュアル ツリー(V)	
60	クイック ウォッチ(O)	Shift+F9	-	ブイブ ブロバブイ エクスブローフー(F) (F)	C1 40
	ブレークポイントの設定/解除(G)	FQ	=	呼()出し腹壁(C) フレービ(U)	Ctrl+Alt+
	ブレークポイントの作成(B)		~	XV9P(H)	Ctrl+Alt
20	すべてのブレークポイントの削除(D)	Ctrl+Shift+F9	-2	モンユール(し)	Ctrl+Alt+
8	すべてのブレークポイントの無効化(N)		φ.	/TUEX(F)	CUITAIC
Ŭ	IntelliTrace(I)		re:	メモリ(M) ※スねいゴル(D)	Carl A Ma
	すべてのデータヒントをクリア(A)		85	レジスタ(G)	Ctrl+Alt+
	データヒントのエクスポート(X)				
1	データヒントのインポート(Y)				
1	名前を付けてダンプを保存(V)				0
-	+======================================				

The below and the ball have all the





① 「デバッグ」 → 「ウインドウ」→「**レジスタ**」



レジスターー

EAX = CCCCCCCC EBX = 00573000 ECX = 00000000 EDX = 001F95C0 ESI = 001F1041 EDI = 006FFDFC EIP = 001F166E ESP = 006FFD30 EBP = 006FFDFC EFL = 00000204

 $0 \times 001 f 9158 = 00000000$ 

#### 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.

Stud	10		
B)	デバ	ッグ(D) チーム(M) Nsight ツール(T)	アーキテクチャ(
		ウィンドウ(W)	
741		グラフィックス(C)	
		続行(C)	F5
00	Ш	すべて中断(H)	Ctrl+Alt+Break
00		デバッグの停止(E)	Shift+F5
	X	すべてデタッチ(L)	
		すべて中止(M)	
	ð	再起動(R)	Ctrl+Shift+F5
	1	コード変更を適用(A)	Alt+F10
	2	パフォーマンス プロファイラー(F)	Alt+F2
	é <sup>io</sup>	プロセスにアタッチ(P)	Ctrl+Alt+P
		その他のデバッグターゲット(H)	
		プロファイラー	
	÷	ステップイン(1)	F11
	?	ステップ オーバー(O)	F10
	:	ステップ アウト(T)	Shift+F11
	2	コードマップに呼び出し履歴を表示(K)	Ctrl+Shift+`





# ① 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.





# ① 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.











