

ca-4. アセンブリ言語

(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





アウトライン



4-1 アセンブリ言語とは 4-2 逆アセンブル

4-3 Visual Studio のデバッガー

4-4 変数のアドレスの確認



4-1 アセンブリ言語とは





・アセンブリ言語とは人間が理解しやすいように、 マシン語(機械語)を翻訳した言語 ・アセンブリ言語の命令と、マシン語の命令が、ほぼ1対1に対応

 4D
 5A
 90
 00
 03
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 00
 <td

実行型ファイルの例

E:¥>ndisa	sm WINWORD.EXE	more	
00000000	4D	dec bp	
00000001	5A	pop dx	
00000002	90	nop	
00000003	0003	add [bp+di],al	
00000005	0000	add [bx+si],al	
00000007	0004	add [si],al	
00000009	0000	add [bx+si],al	
0000000B	00FF	add bh,bh	
0000000D	FF00	inc word [bx+si]	
0000000F	00B80000	add [bx+si+0x0],bh	
00000013	0000	add [bx+si],al	
00000015	0000	add [bx+si],al	
00000017	004000	add [bx+si+0x0],al	
0000001A	0000	add [bx+si],al	
0000001C	0000	add [bx+si],al	
0000001E	0000	add [bx+si],al	
00000020	0000	add [bx+si],al	
00000022	0000	add [bx+si],al	
00000024	0000	add [bx+si],al	
00000026	0000	add [bx+si],al	
00000028	0000	add [bx+si],al	
00000024	0000	add [bx+si].al	

アセンブリ言語の例

Pentium 糸列フロセッサでの アセンブリ言語の主な命令



種類	命令	意味
データ転送と 実効アドレス	MOV	データ転送 ※ ロード,ストア,プッシュ, ポップ
	LEA	実効アドレスのロード
算術演算	ADD	加算
	SUB	減算
	IMUL	乗算
	IDIV	除算
	SAR, SAL	算術シフト
論理演算	AND	論理積
	OR	論理和
	SHR, SHL	論理シフト
比較	CMP	比較
	TEST	AND による比較
ジャンプ(分	JMP	無条件ジャンプ(無条件分岐)
岐)	J??	条件ジャンプ(条件分岐)
サブルーチン	CALL	サブルーチン呼び出し(サブルーチンコール)
	RET	サブルーチンからの復帰



4-2 逆アセンブル





マシン語をアセンブリ言語に翻訳すること









Visual Studioでの逆アセンブル手順





「デバッグ」

「ウインドウ」

→「逆アセンブル」

 $(\mathbf{1})$





逆アセンブルの結果の例

元のマシン語での

アドレス



edi,[ebp-ODOh] 00F117BC lea. 00F117C2 ecx,34h MOV. INNE11707 eax,000000000 MOV. OOF117CC rep stos dword ptr es:[edi] double d = 2.0;00F117CE movsd mmword ptr [d],xmmO 00F117D6 movsd printf("%f, %f ¥n", d, sqrt(d)); 00F117DB sub. esp,8 00F117DF movsd xmmO,mmword ptr [d] 00F117E3 movsd mmword ptr [esp],xmmO _sqrt (OF112A3h) 00F117F8 call 00F117FD fstp qword ptr [esp] 00F117F0 sub. esp,8 00F117F3 xmmO,mmword ptr [d] movsd mmword ptr [esp],xmmO 00F117F8 movsd offset string "%f, %f ¥n" (OF16B3Oh) 00F117FD push. printf (OF11325h) 00F11802 call 00F11807 esp.14h add return O;

アセンブリ言語によるプログラム

アセンブル



- アセンブリ言語を
- ・マシン語に翻訳すること
- ・ (逆アセンブルの逆)





UUF117BC -	lea.	edi,[ebp-UDUh]
00F117C2	MOV	ecx,34h
00F117C7	MOV	eax,OCCCCCCCh
00F117CC	rep stos	dword ptr es:[edi]
double	e d = 2.0;	
00F117CE	movsd	xmm0,mmword ptr [real@4000000000000000000000000000000000000
00F117D6	movsd	nnword ptr [d],xmm0
printf	f(‴%f, %f ¥n'	″, d, sqrt(d));
00F117DB	sub	esp,8
00F117DE	movsd	xmm0,mmword ptr [d]
00F117E3	movsd	mmword ptr [esp],xmmO
00F117E8	call	sgrt (OF112A3h)
00F117ED	fstp	gword ptr [esp]
00F117F0	sub	esp,8
00F117F3	movsd	xmm0,mmword ptr [d]
00F117F8	movsd	mmword ptr [esp],xmmO
00F117FD	push	offset string "%f, %f ¥n" (OF16B3Oh)
00F11802	call	printf (OF11325h)
00F11807	add	esp,14h
returr	n D:	

マシン語による プログラム

アセンブリ言語による プログラム



4-3 Visual Studio の デバッガー





デバッガーとは、プログラムの不具合(バグ)の
 発見や修正を支援するソフトウェアのこと

※ バグを自動的に取り除いてくれるわけではない

<u>主な機能</u> ◆ ブレークポイント機能: プログラムの実行中断 ◆ トレース機能: プログラム実行中に変数の値などを表示

Visual Studio でのブレークポイント設定手順で



ブレークポイントとは、 プログラムの実行を中断 させたい行.複数設定可 ブレークポイントを<u>解除</u>したい ときは赤丸をクリック ※ 同様の操作で、ブレークポ イントの<u>設定</u>もできる

Database Lab.

Visual Studio でのデバッガー起動手順





呼び出し履歴 型 名前 double ConsoleApplication1.exe!m [外部コード] 「下のフレームは間違っている 形が変化! これは、プログ <u>ラムの実行</u>が、ここで中断し ていることを示す

→
② main()

- 🕨 続行(C) - 🎜 🔤 🗉 🔳 🌢 🖄

⊿1∿>Ւ

100

100

▲ プロセス メモリ

- 🔻 対

ゆ ツールの選択 ▼ | ④ ズ-

診断セッション: 0 秒 (249 ミ

▲ CPU (% (全プロセッサに占

デバッガーのステップオーバー機能



デバッガーのステップオーバー機能とは、

ブレークポイントで、プログラム実行が<u>中</u> 断しているときに、<mark>プログラム実行を1行</mark> <u>進める</u>こと





・変数の値の変化

※変数名と値の対応表



• メモリ中のデータの変化

※ダンプリスト形式

・プロセッサのレジスタ内のデータの変化

※ レジスタ名と値の対応

ブレークポイントで中断されたプログラム実 行を再開する手順



変数の値の変化





※「static int x, y, z;」の「static」は、変数 x, y, z のメモリへの格納法を コントロールするキーワード





- 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 main()





・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

	Ŀл	∠ド(B)	デバッグ(D)	チーム(M)	ツール	(T)	₹
]_	Ē.	-בעע	ションのビル	^к (В)		Ctrl+	Sh
	-	291-	ションのソビ	VI~(N)			
-		-בעע	-ションのクリ-	-ン(C)			
$\overline{\mathcal{V}}$		-בעע	-ションでコー	ド分析を実行	(Y)	Alt+F	-11
	*	Console	Application4 0	のビルド(U)			

出力	
出力元(S): ビルド	- 1
- 1> ビルド開始: プロジェクト:ConsoleApplication6, - 1> stdafx.cpp - 1> ConsoleApplication6.cpp	構成:Debug Win32
1> ConsoleApplication6.vcxproj -> e:¥documents¥visual 1> ConsoleApplication6.vcxproj -> e:¥documents¥visual ====================================	studio 2015¥Projectsà studio 2015¥Projectsà ペキップ == <mark></mark> =======



Visual Studioで「x=3;」の行に、ブレークポイン トを設定しなさい





• Visual Studioで, デバッガーを起動しなさい.



- 「x=3;」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





• 「x=3;」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.

ŦΙ	ッグ(D) = - (」(M) Nsight ツール	(T) アーキテクチャ(C)	77	.ト(S) 分析(N) ウィンドウ(W)	ヘルプ(H)	
	942F9(W)			ブレークボイント(B)	Ctrl+Alt+B	
1	())))))(())		 	例外の設定(X) 山力(O)	Ctrl+Alt+E	ŀ
	た(1)(し)	Ctrl+Alt+Rreak		本ガ(0) 診断心(肌の実子(五)	Ctrl + Alt + E2	
	デバッグの停止(5)	Shift+E5			CUITAICTIZ	
×	すべてデタッチ(1)	Shirterio	~	GPU XV%P(U)		22
	すべて中止(M)			ダメク(S)	Ctrl+Shift+D, K	??
ð	再起動(R)	Ctrl+Shift+F5	[:=	型列スタック(K)	Ctri+Snitt+D, S	22
νΞ	□-ド変更を適用(A)	Alt+F10		型 5 1 J 7 7 T (K)		22
	パフォーマンス プロファイラー(F)	Alt+F2	_	リオッナ(W)	Chilli Alberty A	??
d ^{ar}	プロセスにアタッチ(P)	Ctrl+Alt+P	1		Ctrl+Alt+V, A	/? ??
14 14	その他のデバッグ ターゲット(H)				Ctrl+Alt+I	22
e.	プロファ イ ラー			Python Debug Interactive	Shift+Alt+I	20
*	ステップ イン(L)	F11	П	JavaScript コンソール(J)	Ctrl+Alt+V. C	
?	ステップ オーバー(O)	F10		DOM Explorer		
1	ステップ アウト(T)	Shift+F11	0	ライブ ビジュアル ツリー(V)		9 (5
A	コード マップに呼び出し履歴を表示(K)	Ctrl+Shift+`	o	ライブ プロパティ エクスプローラー(P)		
⇔	クイック ウォッチ(Q)	Shift+F9	Œ	呼び出し履歴(C)	Ctrl+Alt+C	
	ブレークポイントの設定/解除(G)	F9	100	スレッド(H)	Ctrl+Alt+H	F
	ブレークポイントの作成(B)			モジュール(O)	Ctrl+Alt+U	
8	すべてのブレークポイントの削除(D)	Ctrl+Shift+F9	1 ⁴⁰	プロセス(P)	Ctrl+Alt+Z	
0	すべてのブレークポイントの無効化(N)			メモリ(M)	÷	
	IntelliTrace(I)		Ģ	逆アセンブル(D)	Ctrl+Alt+D	2 yt
	すべてのデータヒントをクリア(A)		B X	レジスタ(G)	Ctrl+Alt+G	ľ



ローカル いいい いいいいいいいい	
名前	值
G _⊥ x	0
ፍ y	0
🗲 Z	0
② 変数名と	-値の対応表が

表示される

※次ページに拡大図







•「x=3;」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.

F/(→ + + + + + + + + + + + + +	「ワ(D) チー/」M) Nsight ツールC ワインPら(W) ゲロー・シャック 様式 またて、サムマート 様行(C) すべて中断(H) デバッグの停止(E) すべて中断(H) デバッグの停止(E) すべて中断(R) ロード変更を適用(A) 1(フォーマンスプロファイラー(F) Turtar z= 200 - 400) アーキテクチャ(C) F5 Ctrl+Alt+Break Shift+F5 Ctrl+Shift+F5 Alt+F10 Alt+F2 Ctrl+D	77 ■ ■ ■ = = = = = = = = = = = = =	 ト(S) 分析(N ブレークポイント 何外の設定(X) 出力(O) 診断ツールの表 GPU スレッド(U タスク(S) 並列スタック(K 並列ウオッチ(R) ウオッチ(W) 自動変数(A)) ウインドウ(W) (B)) (用) (用))	<pre>\/\J/J(H) Ctrl+Alt+B Ctrl+Alt+FE Ctrl+Alt+F2 Ctrl+Alt+F2 Ctrl+Shift+D, K Ctrl+Shift+D, K Ctrl+Shift+V, A </pre>	2? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??
?' @'' ?' ?'	プロセスにアラッフ(P) その他のデバッグターゲット(H) プロファイラー	Ctri+Ait+P		ローカル(L) イミディエイト(I) Python Debu	g Interactive	Ctrl+Alt+V, L Ctrl+Alt+I Shift+Alt+I	17 77 17 77 17 77 17 77
* ? *	ステップ イン(L) ステップ オーバー(O) ステップ アウト(T)	F11 F10 Shift+F11	5	JavaScript コン DOM Explore	ツール(J) r レッリー(V)	Ctrl+Alt+V, C	e 9 (53
27.	コードマップに呼び出し履歴を表示(K)	Ctrl+Shift+`	0	ライブ プロパティ	ィエクスプローラー(P)		
6 0	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作成(B) すべてのブレークポイントの削除(D)	Shift+F9 F9 Ctrl+Shift+F9		呼び出し履歴(スレッド(H) モジュール(O) ゴロセフ(D)	0	Ctrl+Alt+C Ctrl+Alt+H Ctrl+Alt+U Ctrl+Alt+U	
ି	すべてのブレークポイントの無効化(N)	2	Ø.,			Cur+Alt+Z	
	IntelliTrace(I)		Ģ	逆アセンブル(D)	Ctrl+Alt+D	ミッサに
	すべ (のテーダビントをクリア(A) ニ タビントのエクスポートのの		1	· · · · · · ·		Carriero	_

① 「デバッグ」→ 「ウイン ドウ」→「**逆アセンブル**」

Ě	アセンブル 中	× ConsoleAp	plication7.cpp
<i>7</i>	パドレス(A): ma	ain(void)	
Ŷ) 表示オプショ	ע	
	00CA165C 00CA1662 00CA1667 00CA166C statio	lea mov mov rep stos c int x. v. :	edi,[ebp-OCOh] ecx,30h eax,0CCCCCCCCh dword ptr es:[edi] z:
D	x = 3 00CA166E	, mov	-, dword ptr [x (OCA9138h)],3
	00CA1678	mov	dword ptr [y (OCA913Ch)],4
	Z = X 00CA1682 00CA1687 00CA168D	+ y; mov add mov	eax,dword ptr [x (OCA9138h)] eax,dword ptr [y (OCA913Ch)] dword ptr [z (OCA9140h)],eax
	return 00CA1692 }	n O; xor	eax,eax
	00CA1694 00CA1695 00CA1696	pop pop pop	edi esi ebx
	00CA1697 00CA1699	mov pop	esp,ebp ebp





ステップオーバーの操作を1回ずつ行いながら, 変数 x, y, z の値の変化を確認しなさい.





値
0
0
0

ローカル		
名前		値
Solution	x	3
<	У	0
Solution	Z	0

ローカル		
名前		値
Solution	x	3
<	у	4
Solution	Z	0
ローカル		
名前		値
0	×	2

治則	1旦.
ፍ x	3
ፍ у	4
ፍ z	7



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。





4-4 変数のアドレスの確認



・ソースファイルをそのまま使う.

```
⊟// ConsoleApplication2.cpp :
       11
23
4
5
       #include "stdafx.h"
6
7
      ⊡int main()
8
        ł
9
            static int x, y, z;
10
            x = 3;
                                  ←そのまま使う
            y = 4;
12
            z = x + y;
            return 0;
13
15
```



・ブレークポイントの設定もそのまま使う.

「x = 3;」の行にブレークポイントが設定されていること再確認しておく



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.



- 「x=3;」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





•「x=3;」の行で、実行が中断した状態で、逆アセンブルを行ってみなさい.

tudio) デバ	ッグ(D) チーム(M) Nsight ツール	(T) アーキテクチャ(C)	テスト(S) 分析(N) ウィンドウ(W)	ヘルプ(H)
т #1	ウィンドウ(W) クラノイツクス(C)	> >	 ブレークポイント(B) 例外の設定(X) 単本(2) 	Ctrl+Alt+B Ctrl+Alt+E
	転11(C) すべて中断(H) デパッグの停止(E)	F5 Ctrl+Alt+Break Shift+F5	 (D) (D)	Ctrl+Alt+F2
×	すべてテタッチ(L) すべて中止(M) 再起動(R)	Ctrl+Shift+F5	 	Ctrl+Shift+D, K Ctrl+Shift+D, S ⊧
	コート変更を適用(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) その他のデパッグ ターゲット(H)	Alt+F10 Alt+F2 Ctrl+Alt+P ▶	ウォッチ(W)	► Ctrl+Alt+V, A Ctrl+Alt+V, L Ctrl+Alt+I
	プロファイラー	•	Python Debug Interactive	Shift+Alt+I
• •	ステップ オン(L) ステップ オーバー(O) ステップ アウト(T)	F11 F10 Shift+F11	JavaScript コンソール(J) DOM Explorer	Ctrl+Alt+V, C ▶
<u>1</u>	コードマップに呼び出し履歴を表示(K)	Ctrl+Shift+`		
67	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作成(B)	Shift+F9 F9	 ・呼び出し履歴(C) ・ ・<td>Ctrl+Alt+C Ctrl+Alt+H Ctrl+Alt+U</td>	Ctrl+Alt+C Ctrl+Alt+H Ctrl+Alt+U
* 6	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無効化(N)	Ctrl+Shift+F9		Ctrl+Alt+Z
	Intelli Irace(I) すべてのデータヒントをクリア(A)	•	編 逆アセンブル(D) 図 レッスタ(G)	Ctrl+Alt+D Ctrl+Alt+G

① 「デバッグ」→ 「ウイン ドウ」→「**逆アセンブル**」

UUZƏIbbU rep stos awora ptr es:[eal] static int x, y, z; x = 3;O028166E mov dword ptr [x (0289138h)],3 y = 4; dword ptr [y (028913Ch)],4 00281678 mov z = x + y;00281682 eax,dword ptr [x (0289138h)] MOV 00281687 eax,dword ptr [y (028913Ch)] add dword ptr [z (0289140h)],eax 0028168D mov return O;



逆アセンブルの結果の見方





Visual C++ に対応した アセンブリ言語プログラム



•「x = 3;」の直下に、xのアドレスが表示されているので確認しなさい.後で使う。

例) アドレスは 0289138h

```
UUZԾIbbU rep stos - awora ptr es:[eal]
      static int x, y, z;
      x = 3;
                        dword ptr [x (0289138h)],3
🔾 0028166E - mov
      y = 4;
                        dword ptr [y (028913Ch)],4
  00281678 mov
      z = x + y;
  00281682 mov
                        eax,dword ptr [x (0289138h)]
                        eax,dword ptr [y (028913Ch)]
  00281687 add
  0028168D mov
                        dword ptr [z (0289140h)],eax
      return O;
```



•「x = 3;」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.

ビルド(B)	デバ	ッグ(D) チーム(M) Nsight ツール(T) アーキテクチャ	C) 7:	スト(S) 分析(N) ウィンドウ(W)	ヘルプ(H)	
Debug		ウィンドウ(W)		• 5	」 ブレークポイント(B)	Ctrl+Alt+B	M 🕯 🦄 🦄
51791		ソフノ1ツワス(C)		۱ (F] 例外の設定(X)	Ctrl+Alt+E	· _
	•	続行(C)	F5	•	→ 出力(O)		•
	Ш	すべて中断(H)	Ctrl+Alt+Breal		】 診断ツールの表示(T)	Ctrl+Alt+F2	
	•	デバッグの停止(E)	Shift+F5	p	GPU スレッド(U)		Q 7-612
	х	すべてデタッチ(L)		ń	タスク(S)	Ctrl+Shift+D. K	少(40 ミリ秒 選択)
		すべて中止(M)		6	並列スタック(K)	Ctrl+Shift+D, S	200 - 91
¥projects ール アブ	ð	再起動(R)	Ctrl+Shift+F5	-	並列ウォッチ(R)	· · · ·	
	1	コ−ド変更を適用(A)	Alt+F10		ウォッチ(W)	•	-
		パフォーマンス プロファイラー(F)	Alt+F2	J	自動変数(A)	Ctrl+Alt+V. A	
	¢ [©]	プロセスにアタッチ(P)	Ctrl+Alt+P	[%	」 ローカル(L)	Ctrl+Alt+V, L	
		その他のデバッグターゲット(H)		۱×	1 イミディエイト(I)	Ctrl+Alt+I	
		プロファイラー		۱. ۲۳	Python Debug Interactive	Shift+Alt+I	
	*	ステップ イン(L)	F11	П	JavaScript コンソール(J)	Ctrl+Alt+V. C	2ツリに古める割谷
	?	ステップ オーバー(O)	F10		DOM Explorer	÷	
	:	ステップ アウト(T)	Shift+F11	Ø	ライブ ドジュアル ツリー(い)		
]	۶.	コードマップに呼び出し履歴を表示(K)	Ctrl+Shift+`	0	 ライブ プロパティ エクスプローラー(P) 		
h	⇔	クイック ウォッチ(Q)	Shift+F9	Œ	呼び出し履歴(C)	Ctrl+Alt+C	
[edi]		ブレークポイントの設定/解除(G)	F9	20	スレッド(H)	Ctrl+Alt+H	
(02891385		ブレークポイントの作成(B)		> ⊡	モジュール(O)	Ctrl+Alt+U	
(020010011	8	すべてのブレークポイントの削除(D)	Ctrl+Shift+F9	đ	プロセス(P)	Ctrl+Alt+Z	
(U28913Ch	0	すべてのブレークポイントの無効化(N)		- 6	X王J(M)		
[x (0289		IntelliTrace(I)		•	モアセノフル(D)	Ctrl+Alt+D	豊 メモリ2(2)
(0289140h		すべてのデータヒントをクリア(A)		0.0] レジスタ(G)	Ctrl+Alt+G	二 パビリ 2(2)
		データビントのエクスポート(X)		1-			第 パビリ 5(5)



XEU 1					•••••••••••••••••••••••••••••••••••••••	φ×
アドレス: 0x00281650				- 🖒 列:	0 2 5	
#00001101 #0000100 55 @e e 61 e e 61 00 00 00 03 55 65 78 dei d 40 ff ff ff 14 30 00 00 00 ll c c c c c c f 3 40 e 7 16 38 91 22 00 05 00 #0000100 #0 00 e 1 3 99 12 00 03 50 56 91 26 95 e 40 40 40 ff ff 14 90 30 00 00 ll c c c c c c c c c 1 40 e 7 16 e	00 0 4c 9 cc c 89 3 e8 8 ff f fcc c ff 8 55 8	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	U	SVW - 2050. (.10 - (.39 	27777 + 72.8 · ([· ·] 7777777.2 · (u 777777777.2 · (u · · · · · · · · · · · · · · · · · · ·	2.<
速アセンブル × X ConsoleApplication2.cpp	+	診断ツール			•	Ψ×
PFUZ(A): main(void)	*	☆ ツールの温沢 ▼ ③ フ 	スームイン 🤍 スーム	アウト 🏭 ビューのりも	571	
 ・ ・ ・		診断セッション:0秒(40ミ)	リ砂 選択済み)			-
0020164E Int 3	*		200ミリ秒	400ミリ秒	600ミリ秒	
et¥decuments¥visual studio 2015¥projects¥consoleapplication2¥consoleapplication2¥consoleapplication2+cpp		4 1 K2h				
// ConsoleApplication2.cpp : コンソール アプリケーションのエントリ ホイントを定義します。						





「アドレス」のところに「0289138h」のように記入。「0289138h」は、<u>手順6で調べた値</u>を転記. <u>末尾に「h」</u>

<u>メモリ 1</u>		,
アドレス: 0289138h		
0x00281650 55 8b ec 81 ec c0 00 00 00 5	0x00289166 15 1d 89 81 15 1d 89 81 15 1d	89 81 15
0x002816AC cc cc cc cc b8 eb 10 28 00 c	♥ (
	変数 x, y, z の値はすべて	0
$3 \overline{r} \overline{k} \overline{r} \overline{r} \overline{r} \overline{r} \overline{r} \overline{r} \overline{r} r$		
	(4) メモリの中身が表示され	.ろ
1」りして Enter ナーを押9		
	「x = 3;」は 未実行 で	あ

ることを確認!



• 1 回だけステップオーバーの操作を行い、変数 x の値が 3 に変化することを確認しなさい.





1															
	アドレス:	0x002	2891	38	Х			•	V			Ζ			
	0x00289	138	03	00	00	00	00	00	00	00	00	00	00	00	00
	0x00289	166	15	1d	89	81	15	1 d	89	81	15	1d	89	81	15

変数 x の値が 3 に変化!

「x = 3;」が実行された



もう1回だけステップオーバーの操作を行い、変数yの値が4に変化することを確認しなさい.





さらに3回、ステップオーバーの操作を行い、変数zの値が7に変化することを確認しなさい.





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、デバッガが終了する。







・プログラム

x = 3; 00CA166E mov y = 4; 00CA1678 mov z = x + y; 00CA1682 mov 00CA1687 add 00CA168D mov return 0;







Visual C++ の プログラム



アセンブリ言語







zのアドレス





