

ca-7. データ転送命令と アドレッシングモード

### (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



7-1 データ転送命令 7-2 アドレッシングモード

7-3 配列

7-4 C/C++ の配列は, メモリにどのように格納され ているか



### 7-1 データ転送命令





種類	命令	意味
データ転送と実	MOV	データ転送 ※ ロード, ストア, プッシュ, ポップ
効アドレス	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

プログラムの例



```
#include "stdafx.h"
⊡int_tmain(int_argc, TCHAR* argv[])
  Ĺ
     int a, b;
     a = 10;
     b = a + 20;
     printf("%d, %d¥n", a, b);
     return 0;
     C:¥WINDOWS¥system32¥cmd.exe
       -30
    続行するには何かキーを押してください . .
```

5

### C++ 言語とアセンブリ言語



### Visual C++ の プログラム

int a, b;				
a = 10;				
b = a + 20;				
printf( <mark>″%d</mark> ,	%d¥n″,	a,	b);	
return O;				



--- e:¥documents¥visual studio 2013¥projects¥consoleapplicat #include "stdafx.h"

	int _tmain	n(int argo, _	_TCHAR* argv[])
	00F313C0 00F313C1 00F313C3 00F313C3 00F313CA 00F313CA 00F313CC 00F313D2 00F313D2 00F313D2 00F313D2 00F313DC int a,	push mov sub push push lea mov rep stos b;	ebp ebp,esp esp,DD8h ebx esi edi,[ebp-OD8h] ecx,36h eax,OCCCCCCCCh dword ptr es:[edi]
	a = 10 DOE313DE	J; MOV	dword ptr [a].NAb
	b = a	+ 20;	
~	00F313E5	MOV	eax,dword ptr [a]
	00F313EB	MOV	dword ptr [b],eax
	printf	(‴%d, %d¥n″,	, a, b);
	00F313EE 00F313E0	MOV	esi,esp eax.dword ptr [b]
	00F313F3	push	eax
	00F313F4	MOV	ecx,dword ptr [a]
	00F313F7 00F313F8 00F313FD 00F31403 00F31406 00F31408 return	push call add cmp call o 0;	ecx OF35858h dword ptr ds:[OF39114h] esp,OCh esi,esp RTC_CheckEsp (OF31140h)
	00F3140D	xor	eax,eax
	) DOF3140F 00F31410 00F31411 00F31412 00F31418 00F3141A 00F3141F 00F31421 00F31422 \7 7 7	pop pop add cmp call mov pop ret ファイルがi	edi esi ebx esp,0D8h ebp,esp RTC_CheckEsp (OF31140h) esp,ebp ebp ありません
			J
		T	
			オペリンド
		כד נ	ハットノノト

6





#### 「データ転送」せよの命令 ・メモリからデータを読み出して、レジスタに書き込み ・レジスタからデータを読み出して、メモリに書き込み ・レジスタからデータを読み出して、別のレジスタに書き込み など

e:¥documents¥vi #include ″stdafx.h^	isual studio 2013¥projects¥con ″	soleapplicat	int a	la e	
int _tmain(int argo	c, _TCHAR∗ argv[])		int a	, W,	
00F313CO push 00F313C1 mov 00F313C3 sub 00F313C3 sub 00F313C9 push 00F313CA push	ebp ebp,esp esp,008h ebx esi		a = 10	);	
00F313CB push 00F313CC lea 00F313D2 mov 00F313D7 mov 00F313D7 mov	edi edi,[ebp-OD8h] ecx,38h eax,0CCCCCCCCh dmord ptr col[edi]		00F313DE	MOV	dAO.[s] rta browb
int a, b; a = 10; OOF313DE mov b = a + 20; ODE913EE mov	dword ptr [a],OA		b = a	+ 20;	メモリからレジスタ eax へ
00F313E8 add 00F313E8 moy printi wd, wd* 00F313EE moy	eax,14h dword ptr [b].eat en , &, D); esi,esp		00F313E5	MOV	eax,dword ptr [a]
00F313F3 push 00F313F4 mov 00F313F4 mov 00F313F7 push 00F313F8 push	eax, dword ptr [b] eax ecx, dword ptr [a] ecx OF35858h		00F313E8	add	eax,14h
00F31403 add 00F31406 cmp 00F31406 call return 0;	esp,OCh esi,esp RTC_CheckEsp (OF31140h)		00F313EB	MOV	dword ptr [b],eax
<pre>&gt; D0F3140D xor &gt; D0F3140F pop 00F31410 pop 00F31411 pop 00F31412 add 00F31412 add</pre>	eax,eax edi esi ebx esp,0D8h		nrinti	F (11%d).	<sup>※付:</sup> レジスタ eax からメモリへ
00F31410 CMP 00F3141A call 00F3141F mov 00F31421 pop 00F31422 ret ソーフ ファイル	eop,esp RTC_CheckEsp (OF31140h) esp,ebp ebp				
				`` <u>/_</u> -	

アセンブリ言語のプログラム

テータ転送命令 mov 命令 が使用されている



### 7-2 アドレッシングモード

アドレッシングモードのバリエーション







メモリへの書き込み, メモリからの読み出し を行う **アドレッシングモード**  アドレッシングモードのバリエーション







<u>値</u>を扱う アドレッシングモード

アドレッシングモードのバリエーション



#### Visual C++ の アセンブリ言語 プログラム





<u>レジスタ</u>を扱う アドレッシングモード





```
次のプログラムを Visual Studio で実行し,
  結果を確認しなさい
⊟int main()
     int a:
     asm {
                             ①レジスタEAXに値10をセット
         mov eax, 10;
                             ②レジスタEAXに20を足しこむ
         add eax, 20;
                             ③レジスタEAXの値を,<u>変数a</u>のア
         mov a, eax;
     };
                             ドレスに書き込む
     printf("a = %d", a);
     return 0;
   Studio
                                     [a = 30]
                       C:¥WINDOWS¥s
    ビルド(B)
       デバッグ(D) チーム(M)
              Nsight ツール(T
   Debug
        ウィンドウ(W)
```





が表示されたら成功



### 7-3 配列

C/C++ での配列と繰り返し



```
∃int main()
     static int x[5] = { 8, 6, 4, 2, 3 };
     static int y[5] = \{0, 0, 0, 0, 0\};
     int i:
     for (i = 0; i < 5; i++) {
         y[i] = x[i] * 10;
                            繰り返す処理
     return O:
```

i の値は 0, 1, 2, 3, 4 と<u>変化</u>し, 全部済んだら終わる





- 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 #include "stdafx.h"





・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	レド(B)	デバッグ(D)	チーム(M)	ツール(	(T) <del>,</del>
1-	Ē.	-בעע	-ションのビル	≺́(В)		Ctrl+Sh
	-	291-	ションのソビ	UT(K)		
-		-בעע	-ションのクリ-	-ン(C)		
$\overline{\mathcal{V}}$		-בעע	-ションでコート	ド分析を実行	(Y)	Alt+F11
	*	Console	Application4 0	Dビルド(U)		

出力	
出力元(S): ビルド	• 🖆 🖆 🛓
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 <del>studio 20</del> 15¥Projectsà ミキップ == <mark></mark> =======



# Visual Studioで「int i;」の行に、ブレークポイン トを設定しなさい



(E	デノ	(ッグ(D)ム(M) ツー	JL(T)	テスト(S)	□ \ 分析(I
ndov		ワイントワ(W) グラフィックス			
		デバッグ開始(S)		F5	
1	₽	デバッグなしで開始(H)		Ctrl+	F5
	e <sup>®</sup>	プロセスにアタッチ(P)			
		例外(X)		Ctrl+	Alt+E
		パフォーマンスと診断(F)		Alt+F	2
	ς.	ステップ イン(I)		F11	
	ß	ステップ オーバー(0)		F10	
		ブレークポイントの設定/解除	(G)	F9	
		フレージバインドのYFAX(B)			
	1	すべてのブレークポイントの	削除(D)	Ctrl+	Shift+F
		すべてのデータヒントをクリ!	ア(A)		
		データヒントのエクスポート	'X)		







ントが設定される

赤丸がブレークボ

イントの印

ので確認.



• Visual Studioで, デバッガーを起動しなさい.



- 「int i;」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 i;」の行で、実行が中断した状態で、<u>変数の値</u> <u>を表示</u>させなさい、手順は次の通り、





20



ローカル	
名前	値
🥥 i	-858993460
Þ 😪 🗙	0x00229000 {8, 6, 4, 2, 3}
🕨 📬 y	0x00229150 {0, 0, 0, 0, 0}



### ・「int i;」の行で,実行が中断した状態で,<u>逆アセン</u> ブルを行いなさい.

	ック(0) 7-12 M) Nsight アール(1, ウインドク(W) グラス・トライク	)-459574(C)	•	r(3) プロイバ(N) ブレークポイント(B) 例外の設定(X)	912179(W)	Ctrl+Alt+B Ctrl+Alt+E	
?" ?" ?" ?" ?" ?" ?" ?" ?" ?" ?"	続行(C) すべて中断(H) デパッグの停止(E) すべてデタッチ(L) すべて中止(M) 再起動(R) コード変更を遮用(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)	F5 Ctrl+Alt+Break Shift+F5 Ctrl+Shift+F5 Alt+F10 Alt+F2 Ctrl+Alt+P ↓		出力(0) 診断ツールの表示(1 GPU スレッド(U) ダスク(5) 並列スタック(K) 並列カッチ(K) 自動変数(A) ローカル(L) イミディエト(1)	ד)	Ctrl+Alt+F2 Ctrl+Shift+D, K Ctrl+Shift+D, S Ctrl+Alt+V, A Ctrl+Alt+V, L Ctrl+Alt+I	??????????????????????????????????????
° + ? + ૠ ↔	プロファイラー ステップ イン(L) ステップ オーパー(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)	F11 F10 Shift+F11 Ctrl+Shift+`		Python Debug Im JavaScript IDV- DOM Explorer จิกวี ไข้มาไป พ. จิกวี ไข้มาไป พ.	teractive リレ(J) リー(V) クスプローラー(P)	Shift+Alt+I Ctrl+Alt+V, C	) 2 22 ⊕ (53 ≷
€ • •	ジィッジンガマテ(LU)… 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) すべてのブレークポイントの削除(D)	F9 Ctrl+Shift+F9	( x 0 %	呼び出し履歴(C) スレッド(H) モジュール(O) プロセス(P)		Ctrl+Alt+C Ctrl+Alt+H Ctrl+Alt+U Ctrl+Alt+Z	
	IntelliTrace(I) すべてのデータヒントをクリア(A) データヒントのエクスポートハハ		5	逆アセンブル(D)		Ctrl+Alt+D	2ッサに,



	static int x[5] static int y[5] int i:	=	{ 8, 6, 4, 2, 3 }; { 0, 0, 0, 0, 0 };
0	for (i = 0; i < 0022166E mov 00221675 jmp 00221677 mov 0022167A add 0022167D mov	5;	: i++) { dword ptr [i],0 main+30h (0221680h) eax,dword ptr [i] eax,1 dword ptr [i].eax
	00221680 cmp 00221684 jge y[i] = x[i] 00221686 mov 00221689 imul 00221691 mov	*	dword ptr [i],5 main+4Dh (022169Dh) 10; eax,dword ptr [i] ecx,dword ptr x (0229000h)[eax*4],0Ah edx,dword ptr [i]
	00221694 mov } 0022169B jmp return 0; 0022169D vor		aword ptr y (U22915Uh)[edx*4],ecx main+27h (O221677h)





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

۲(B)	デバ	ッグ(D) : ーム(M) Nsight ツール(T)	7
ug 📑		- ワイントワ(W)	
イフサイ		グラフィックス(C)	
		続行(C)	F5
00000000		すべて中断(H)	Ctrl
22 21		デバッグの停止(E)	Shit
?? ?'	×	すべてデタッチ(L)	
????' ????'		すべて中止(M)	
<u>??</u> ?'	ð	再起動(R)	Ctrl
77 7 77 7	<u>_</u>	コ−ド変更を適用(A)	Alt
?? ?'	2	パフォーマンス プロファイラー(F)	Alt
27 2	a <sup>a</sup>	プロセスにアタッチ(P)	Ctrl
?? ?' ?? ?'		その他のデバッグターゲット(H)	
<u></u>		プロファイラー	
	÷	ステップ イン(L)	F11
	?	ステップ オーバー(O)	F10
	:	ステップ アウト(T)	Shit
	ÞŦ	コードマップに応び出し民族をキキハハ	Ctel



אונו ש	
名前	値
🤗 i	-858993460
▷ 😭 x	0x00229000 {8, 6, 4, 2, 3}
▷ 🐔 y	0x00229150 {0, 0, 0, 0, 0}
את ב	
名前	値
🤗 i	0
▷ 🐔 x	0x00229000 {8, 6, 4, 2, 3}
▷ ፍ y	0x00229150 {80, 0, 0, 0, 0}
- vitr -	
名前	値
🤗 i	1
▷ 🗲 x	0x00229000 {8, 6, 4, 2, 3}
Þ ፍ y	0x00229150 {80, 60, 0, 0, 0}
名前	値
- //// 名前 ❷ i	值 2
名前 ② i ▶ € x	值 2 0x00229000 {8, 6, 4, 2, 3}
名前 ② i ▷ ☞ x ▷ ☞ y	値 2 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 0, 0}
名前 ❷ i ▷ � x ▷ � x ▷ � y	値 2 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 0, 0}
名前 ② i ▷ 喻 x ▷ 喻 y □ <i>JJJP</i>	値 2 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 0, 0}
名前 ② i ▷ 喻 x ▷ 喻 y 名前 名前	値 2 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 0, 0} 値 2
名前 ② i ▷ ☜ x ▷ ☜ y 名前 名前 名前 2 JJJP	値 2 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 0, 0} 値 3
名前 ② i ▷ ☜ x ▷ ☜ y 名前 ④ i ▷ ☜ x ▷ ☜ x	値 2 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 0, 0} 値 3 0x00229000 {8, 6, 4, 2, 3} 0x00229000 {8, 6, 4, 2, 3}
名前 ② i ▷ ☜ x ▷ ☜ y 名前 ④ i ▷ ☜ x ▷ ☜ x ▷ ☜ y	値 2 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 0, 0} 値 3 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 20, 0}
名前 ② i ▷ ☜ x ▷ ☜ y 名前 ④ i ▷ ☜ x ▷ ☜ x ▷ ☜ y	値 2 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 0, 0} 値 3 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 20, 0}
名前 ② i ▷ ☞ x ▷ ☞ y 名前 ④ i ▷ ☞ x ▷ ☞ y 名前 ○ i ▷ ☞ x ▷ ☞ y 名前	値 2 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 0, 0} 值 3 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 20, 0} 值
名前 ② i ▷ ☞ x ▷ ☞ y 名前 ④ i ▷ ☞ x ▷ ☞ y 名前 ④ i ▷ ☞ y 名前 ④ i ▷ ☞ i ▷ ☞ y	値 2 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 0, 0} 値 3 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 20, 0} 値 4
名前 ② i ▷ ☞ x ▷ ☞ y 名前 ④ i ▷ ☞ x ▷ ☞ y 乙 ろ前 ④ i ▷ ☞ x ▷ ☞ y 乙 ろ前 ● ☞ x ▷ ☞ y 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇	値 2 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 0, 0} 値 3 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 20, 0} 値 4 0x00229000 {8, 6, 4, 2, 3}
名前 ② i ▷ 全面 x ▷ 全面 y 名前 ④ i ▷ 全面 x ▷ 全面 y 乙 乙 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇	(値 2 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 0, 0} (値 3 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 20, 0} (値 4 0x00229000 {8, 6, 4, 2, 3} 0x00229150 {80, 60, 40, 20, 30}



# ・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.





### 7-4 C/C++ の配列は, メモリにどのように格納され ているか







リトルエンディアンで, 4バイトの数値に**コード化**した場合の例

![](_page_26_Picture_0.jpeg)

![](_page_26_Picture_1.jpeg)

- •C 言語や, C++ 言語の配列は, 同じ型の要素の並び.
- コード化されて、メモリに格納されるとき、
   要素が順にメモリに格納される
   各要素のサイズは同じ

![](_page_27_Picture_0.jpeg)

![](_page_27_Picture_1.jpeg)

- 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

![](_page_28_Picture_0.jpeg)

### 先ほどのプログラムをそのまま使う

#include "stdafx.h"

![](_page_28_Figure_3.jpeg)

![](_page_29_Picture_0.jpeg)

・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	レド(B)	デバッグ(D)	チーム(M)	ツール	(T)	テ
]_	Ē.	-בעע	ションのビル	≺́(В)		Ctrl+	Sh
	-	291	ションのソビ	UT(K)			
-		-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+I	F11
	*	Console	Application4 0	Dビルド(U)			

出力	
出力元(S): ビルド	• 🖆 🖆 🛓
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 <del>studio 20</del> 15¥Projectsà ミキップ == <mark></mark> =======

![](_page_30_Picture_0.jpeg)

# Visual Studioで「int i;」の行に、ブレークポイン トを設定していること

![](_page_30_Figure_2.jpeg)

![](_page_31_Picture_0.jpeg)

• Visual Studioで, デバッガーを起動しなさい.

![](_page_31_Picture_2.jpeg)

- 「int i;」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

![](_page_31_Figure_5.jpeg)

![](_page_32_Picture_0.jpeg)

## 「int i;」の行で、実行が中断した状態で、<u>変数の値</u> <u>を表示</u>させなさい、手順は次の通り、

÷.r	ッグ(D) = -仏(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)
	ウィンドウ(W)		•	ブレークポイント(B)	Ctrl+Alt+B
	クラノイツクス(C)	1	• •	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+Break		診断ツールの表示(T)	Ctrl+Alt+F2
	デバッグの停止(E)	Shift+F5	12	GPU スレッド(U)	
X	すべてデタッチ(L)		Ô	タスク(S)	Ctrl+Shift+D, K
	すべて中止(M)		ćΞ	並列スタック(K)	Ctrl+Shift+D, S
δ	再起動(R)	Ctrl+Shift+F5		並列ウォッチ(R)	÷
<u></u>	コード変更を適用(A)	Alt+F10		ウォッチ(W)	•
	バフォーマンス プロファイラー(F)	Alt+F2			Ctrl+Alt+V, A
0 <sup>84</sup>	プロセスにアタッチ(P)	Ctrl+Alt+P	<b>t</b> #]	ローカル(L)	Ctrl+Alt+V, L
	その他のテバッグターゲット(H)	1			Ctrl+Alt+I
	プロファイラー	1		Python Debug Interactive	Shift+Alt+I
*	ステップ イン(L)	F11	5	JavaScript コンソール(J)	Ctrl+Alt+V, C
3	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)	
Þ.	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)	)
⇔	クイック ウォッチ(Q)	Shift+F9	E	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	72	スレッド(H)	Ctrl+Alt+H
	ブレークポイントの作成(B)			モジュール(O)	Ctrl+Alt+U
ð	すべてのブレークポイントの削除(D)	Ctrl+Shift+F9	e <sup>ja</sup>	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
	すべてのデータヒントをクリア(A)		0X	レジスタ(G)	Ctrl+Alt+G

![](_page_32_Picture_3.jpeg)

o-bh	
名前	値
🥥 i	-858993460
▷ 😭 x	0x00229000 {8, 6, 4, 2, 3}
Þ ፍ y	0x00229150 {0, 0, 0, 0, 0}

![](_page_32_Picture_5.jpeg)

※次ページに拡大図

![](_page_33_Picture_0.jpeg)

## 「int i;」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。

?' ?' ?' ?' ?' ?' ?' ?' ?' ?' ?'	ウイソトウ(W) 括日、ウロイク 続行(C) すべて中断(H) デパッグの停止(E) すべて中止(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/う(-(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)	F5 Ctrl+Alt+Break Shift+F5 Ctrl+Shift+F5 Alt+F10 Alt+F2 Ctrl+Alt+P		□ フレークボイント(8) □ 例外の設定(X) ■ 出力(O) ■ 診断ツールの表示(T) ■ びロノッド(U) □ クスク(S) ■ 並列スタック(K) 並列スタック(K) ■ クカッチ(R) ○ オッチ(W) ■ 自動変数(A) ■ つカル(L)	Ctrl+Alt+B Ctrl+Alt+E Ctrl+Alt+F2 Ctrl+Shift+D, K Ctrl+Shift+D, S Ctrl+Alt+V, A Ctrl+Alt+V, L	
? * ( ? *	プロファイラー 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			Ctrl+Alt+I Shift+Alt+I Ctrl+Alt+V, C	???? ?? 
₩ ↔ •	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作成(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無効化(N)	Ctrl+Shift+` Shift+F9 F9 Ctrl+Shift+F9	6 7 7 8	<ul> <li>ライブ プロパティエクスプローラー(P)</li> <li>ライブ プロパティエクスプローラー(P)</li> <li>♥ び出し履歴(C)</li> <li>ズレッド(H)</li> <li>Ξ モジュール(O)</li> <li>ジ プロセス(P)</li> </ul>	Ctrl+Alt+C Ctrl+Alt+H Ctrl+Alt+U Ctrl+Alt+Z	
	IntelliTrace()) すべてのデータヒントをクリア(A) データヒントのエクフポート(Y)		Ģ	<ul> <li>ジアセンブル(D)</li> <li>レンハス(の)</li> </ul>	Ctrl+Alt+D	zッサに

![](_page_33_Picture_3.jpeg)

	static int x[5] = static int y[5] = int i; for (i = 0; i < 5	{ 8, 6, 4, 2, 3 }; { 0, 0, 0, 0, 0 }; ; i++) {
0	D022166E mov	dword ptr [i],O
	00221675 jmp	main+30h (0221680h)
	00221677 mov	eax,dword ptr [i]
	0022167A add	eax,1
	0022167D mov	dword ptr [i],eax
	00221680 cmp	dword ptr [i],5
	00221684 jge	main+4Dh (O22169Dh)
	y[i] = x[i] *	10;
	00221686 mov	eax,dword ptr [i]
	00221689 imul	ecx,dword ptr x (0229000h)[eax*4],OAh
	00221691 mov	edx,dword ptr [i]
	00221694 mov	dword ptr y (0229150h)[edx*4],ecx
	}	
	0022169B jmp	main+27h (0221677h)
	return O;	
	10022169D_vor	00V 00V

![](_page_33_Picture_5.jpeg)

![](_page_34_Picture_0.jpeg)

## ・ ローカルウインドウで配列 y の先頭アドレスを調べなさい

名前	值
🤗 i	-858993460
Þ ፍ x	0x00229000 {8, 6, 4, 2, 3}
Þ 🐔 y	0x00229150{0, 0, 0, 0, 0}

### 「0x」が付いているの は16進数

![](_page_34_Picture_4.jpeg)

![](_page_35_Picture_0.jpeg)

### プログラムの中で配列yの先頭アドレスがある ことを確認しなさい

```
static int v[5] = \{0, 0, 0, 0, 0\};
       int i;
      for (i = 0; i < 5; i++) {
OD22166E
                          dword ptr [i],0
             MOV.
                           main+30h (0221680h)
  00221675
             imp.
  00221677
                           eax,dword ptr [i]
             MOV.
  0022167A
             add.
                          eax.1
  0022167D
                          dword ptr [i],eax
             MOV.
  00221680
                           dword ptr [i],5
             CMP.
                          main+4Dh (022169Dh)
  00221684
             ige
           y[i] = x[i] * 10;
  00221686
                           eax,dword ptr [i]
             MOV.
                           ecx,dword ptr x (0229000h)[eax*4],0Ah
  00221689
             imul
                          edx,dword ptr [i]
dword ptr y (0229150h)[edx*4],ecx
  00221691
             MOV.
  00221694
             MOV.
```

「h」が付いているのは 16進数

![](_page_36_Picture_0.jpeg)

### ダンプリストを表示させなさい.

5	デバッグ(D) <sup>E</sup> ーム(M) ツール(T) テスト(S	) 分析(N) ウィン	ドウ	(W) ヘルプ(H)	มีสวสว 🎦	
1	ウィンドウ(W) ガニス (ックス	•	61 Ki	ブレークポイント(B) 出力(0)	Ctrl+Alt+B	
	<ul> <li>         ・</li></ul>	F5 Ctrl+Alt+Break Shift+F5	n Â	GPU スレッド(U) タスク(S) 並列スタック(K)	Ctrl+Shift+D, K Ctrl+Shift+D, S	
) () ()	<ul> <li>▼ マヘてデタッチ(D)</li> <li>▼ マヘて中止(M)</li> <li>● 用起動(R)</li> <li>● プロセスにアタッチ(P)</li> <li>● 例(M(X)</li> </ul>	Ctrl+Shift+F5 Ctrl+Alt+F	, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	並列ウォッチ(R) ウォッチ(W) 自動変数(A) ローカル(L)	Ctrl+Alt+V, A Ctrl+Alt+V, L	
	パフォーマンスと診断(F)	Alt+F2		イミディエイト(I) JavaScript コンソール(J)	Ctrl+Alt+I ) Ctrl+Alt+V, C	
( (	<ul> <li>G. ステップ イン(I)</li> <li>G. ステップ オーバー(0)</li> <li>G. ステップ アウト(T)</li> </ul>	F11 F10 Shift+F11		DOM Explorer(D) 呼び出し履歴(C) スレッド(H)	Ctrl+Alt+V, D Ctrl+Alt+C Ctrl+Alt+H	
6	↔ クイック ウォッチ(Q) ブレークポイントの設定/解除(G)	Shift+F9 F9		モジュール(0) プロセス(P)	Ctrl+Alt+U Ctrl+Alt+Z	
ě	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) すべてのブレークポイントを無効にする(N)	► Ctrl+Shift+F9		メモリ(M) ※マセンゴル(D) レジスタ(G)	rl+Alt+D Ctrl+Alt+G	<ul> <li>         は、メモリ 1(1) Ctrl+A         は、メモリ 2(2) Ctrl+A         は、ハモリ 2(2) Ctrl+A         は、ハモリ 3(3) Ctrl+A         は、ハーモリ 3(3) Ct</li></ul>

![](_page_36_Picture_3.jpeg)

![](_page_36_Picture_4.jpeg)

![](_page_36_Picture_5.jpeg)

![](_page_37_Picture_0.jpeg)

### ・ 配列 y の先頭アドレスを, メモリウインドウの「アドレス」のところに 書き写して, Enter キーを押す

0x00229000 {8, 6, 4,

0x00229150 0, 0, 0,

#### 「<mark>0x</mark>00229150」のように 頭に <mark>0x</mark> を付ける

![](_page_37_Picture_3.jpeg)

□-刀ル 名前 値 ・858993460

1	配列 <u>y の先頭ア</u>
	<u>ドレス</u>

🕨 📬 🗴

🕨 📬 y

![](_page_38_Picture_0.jpeg)

## ・ メモリウインドウに、配列 y の中身が表示されるので確認する

![](_page_38_Picture_2.jpeg)

### 00 が並んでいる

![](_page_39_Picture_0.jpeg)

### ・ステップオーバーの操作を行いながら、メモリの 中身の変化を確認しなさい.

メモリ 1

アドレス: 0x00229150

アドレス: 0x00229150

	デノ	(ッグ(D)	チーム(M)	ツール(T)	テスト(S)	分析(N
٣		ワイント	(W)			
۴		グラフィッ	ックス			
	•	続行(C)				F5
	Ш	すべて中国	所(K)			Ctrl+Alt
7		デバッグの	D停止(E)			Shift+F
	$\mathbf{X}$	すべてデタ	タッチ(D)			
		すべて中止	E(M)			
	ΰ	再起動(R)				Ctrl+Sh
	ø	プロセスに	こアタッチ(P).			
_		例外(X)				Ctrl+Alt
		パフォーマ	マンスと診断(P	F)		Alt+F2
	ς.	ステップ・	イン(I)			F11
	4	ステップ	オーバー(0)			F10
	¢,	ステップ	アウト(T)			Shift+F
	⇔	クイック	ウォッチ(Q)			Shift+F!
		ブレークオ	ポイントの設定	E/解除(G)		F9
		ブレークオ	ポイントの作成	ҟ(B)		

![](_page_39_Picture_3.jpeg)

![](_page_40_Picture_0.jpeg)

### •10 進数にすると, 80, 60, 40, 20, 30 である

#### 16進数 10進数

XEU 1		
.d アドレス: 0x00229150 ■x00229150 00 00 00 00 00 00 00 00 00 00 00 00 0	50	80
XEU 1		
アドレス: 0x00229150		• •
0x00229150 50 00 00 00 00 00 00 00 00 00 00 00 0	30	60
	•••	•••
XEU 1		
4 アドレス: 0x00229150		
- UXUU229150 50 00 00 00 00 00 00 00 00 00 00 00 0	00	40
XEU 1	20	40
アドレス: 0x00229150		
- 0x00229150 50 00 00 00 3c 00 00 00 28 00 00 00 00 00 00 00 00 00 00 00 0		
ХЕ И 1	14	20
4 アドレス: 0x00229150	1-1	20
- 0x00229150 50 00 00 00 3c 00 00 00 28 00 00 14 00 00 00 00 00 00 0 0 0 0 0 0		
		• • •
4 アトレス: 0x00229150 0x00229150 50 00 00 00 3c 00 00 00 28 00 00 00 14 00 00 00 1c 00 00 00 0	1e	- 30
		$\sim$

![](_page_41_Picture_0.jpeg)

# ・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.

![](_page_41_Picture_2.jpeg)

### 配列の要素に値を格納するプログラム

![](_page_42_Picture_1.jpeg)

![](_page_42_Figure_2.jpeg)

### 配列の要素に値を格納するプログラム

![](_page_43_Picture_1.jpeg)

#### Visual C++ では: a[3] = 4

![](_page_43_Figure_3.jpeg)

種々のアドレッシングモード

![](_page_44_Picture_1.jpeg)

b = a + <b>200</b> ;	mov eax,dword ptr ds:[00348130h] add eax,0c8h	0c8h
b = a + <b>x</b> ;	mov eax,dword ptr ds:[00258130h] add eax,dword ptr ds:[258138h]	変数 x のア ドレス
b = a + <b>y[5]</b> ;	<pre>mov eax,4 imul ecx,eax,5 mov edx,dword ptr ds:[1198130h] add edx,dword ptr ds:[ecx+1198138h]</pre>	y[5]があるア ドレス
b = a + <b>y[i]</b> ;	<pre>mov eax,dword ptr ds:[00048160h] mov ecx,dword ptr ds:[48130h] add ecx,dword ptr ds:[eax*4+48138h]</pre>	y[i]があるア ドレス