embarcadero
Developer Camp

【G6】ライトニングトーク

呼出規約の違いを アセンブラレベルで比較する

東洋テクニカルシステム株式会社
システム開発部
福士　光

embarcadero
Developer Camp

アジェンダ

アジェンダ

- 呼出規約とは
- Windows(x86)の呼出規約の比較
- Windows(x86)の呼出規約によるコード生成の比較
- クラスのメソッドの扱い
- Windows(x64)のコード生成

- 詳しい説明は後でセッション資料をダウンロードして
  ご確認ください。

embarcadero
Developer Camp

基本知識

基本知識～呼出規約とは？

- 呼出規約とは？
  - サブルーチンとのインタフェースに関する決まりごと
  - 呼出元(caller)と呼出先(callee)で一致していないと正しく動作しない
  - 一般には
    - パラメータと戻値をどこに置くか
    - パラメータをどの順序で渡すか
    - クリーンアップの責任分担
    - レジスタの使い方
    - 名前の修飾(name mangling)
    - など

embarcadero
Developer Camp

Windows(x86)の呼出規約

Windows(x86)の呼出規約(1)

- Windows(x86)+Delphi上の呼出規約とその主な違い

呼出規約	パラメータを渡す順序	スタック上のパラメータの削除	パラメータの引き渡しにレジスタを使用
register	左から右	呼出先	あり
pascal	左から右	呼出先	なし
cdecl	右から左	呼出元	なし
stdcall	右から左	呼出先	なし
safecall	右から左	呼出先	なし

Windows(x86)の呼出規約(2)

- パラメータを渡す順序が「右から左」であれば可変長引数が容易(Cのprintf関数のように)
  - レジスタまたはスタックの既知の位置にあるパラメータから残りのパラメータの数とサイズを推定できる
- スタック上のパラメータの削除を呼出元で行うと、呼出毎にパラメータのクリーンアップコードが必要
  - 可変長引数では呼出先でクリーンアップすることは困難
- スタック上のパラメータの削除を呼出先で行うと、パラメータのクリーンアップコードが1箇所で済む
  - 16bit Windowsのころはメモリが潤沢ではなかった

Windows(x86)の呼出規約(3)

- Delphiの`register`規約では最大3つの32bit値をレジスタ経由で渡すことができる
  - EAX/EDX/ECXレジスタを使用
  - この範囲であればスタックフレームを作成しなくてすむ
  - それ以上のパラメータはスタック経由
- `stdcall`規約はWin32 APIあるいは標準的なDLLの呼び出しに使用する
- `safecall`規約は`stdcall`に加えて例外「ファイアウォール」を実装している
  - COMのエラーを例外に変換する

Windows(x86)の呼出規約(4)

- 順序型の戻値はAL(8bit)/AX(16bit)/EAX(32bit)/EDX:EAX(64bit)に格納される
- 実数型の戻値はFPUのスタックトップレジスタST(0)に格納される

- 詳細はヘルプで(参考文献を参照)

Windows(x86)の呼出規約(5)

- アセンブルリストはデバッグ実行中に以下の方法で表示
  - メインメニュー→表示→デバッグ→CPUウィンドウ→CPU全体
  - ショートカットキーは[Ctrl]+[Alt]+[C]

- アセンブルリスト上での自動変数領域
  - [ebp-$xx]はローカル変数(x64では[rbp-$xx])
  - [ebp+$xx]は仮引数(x64では[rbp+$xx])

Windows(x86)の呼出規約(6)

- サンプル
  - 引数はInteger x 4、戻値もInteger
  - 呼出規約を変化させて呼出元、呼出先の生成コードを確認してみる
  - デバッグビルド(x86、x64とも)

register規約でも
スタックを使用させるため

```
function Foo(Param1: Integer; Param2: Integer; Param3: Integer; Param4: Integer): Integer;
begin
  Result := Param1 + Param2 + Param3 + Param4;
end;

var
  RetVal: Integer;
begin
  RetVal := Foo(1,2,3,4);
  Writeln(RetVal);
end.
```

Windows(x86)の呼出規約 - register規約

```
CC_Register.dpr.15: RetVal := Foo(1,2,3,4);
004060D4 6A04              push $04
004060D6 B903000000        mov ecx,$00000003
004060DB BA02000000        mov edx,$00000002
004060E0 B801000000        mov eax,$00000001
004060E5 E8E6F1FFFF        call Foo
004060EA A39CAB4000        mov [$0040ab9c],eax

CC_Register.dpr.8: begin
004052D0 55                push ebp
004052D1 8BEC              mov ebp,esp
004052D3 83C4F0            add esp,-$10
004052D6 894DF4            mov [ebp-$0c],ecx
004052D9 8955F8            mov [ebp-$08],edx
004052DC 8945FC            mov [ebp-$04],eax
CC_Register.dpr.9: Result := Param1 + Param2 + Param3 + Param4;
004052DF 8B45FC            mov eax,[ebp-$04]
004052E2 0345F8            add eax,[ebp-$08]
004052E5 0345F4            add eax,[ebp-$0c]
004052E8 034508            add eax,[ebp+$08]
004052EB 8945F0            mov [ebp-$10],eax
CC_Register.dpr.10: end;
004052EE 8B45F0            mov eax,[ebp-$10]
004052F1 8BE5              mov esp,ebp
004052F3 5D                pop ebp
004052F4 C20400            ret $0004
```

第4パラメータだけがスタックに積まれ、残りはEAX/EDX/ECXに格納

戻値を格納

スタックから4バイトをクリアしてリターン

Windows(x86)の呼出規約 - pascal規約

```
CC_Pascal.dpr.15: RetVal := Foo(1,2,3,4);
004060D4 6A01              push $01
004060D6 6A02              push $02
004060D8 6A03              push $03
004060DA 6A04              push $04
004060DC E8EFF1FFFF        call Foo
004060E1 A39CAB4000        mov [$0040ab9c],eax

CC_Pascal.dpr.8: begin
004052D0 55                push ebp
004052D1 8BEC              mov ebp,esp
004052D3 51                push ecx
CC_Pascal.dpr.9: Result := Param1 + Param2 + Param3 + Param4;
004052D4 8B4514            mov eax,[ebp+$14]
004052D7 034510            add eax,[ebp+$10]
004052DA 03450C            add eax,[ebp+$0c]
004052DD 034508            add eax,[ebp+$08]
004052E0 8945FC            mov [ebp-$04],eax
CC_Pascal.dpr.10: end;
004052E3 8B45FC            mov eax,[ebp-$04]
004052E6 59                pop ecx
004052E7 5D                pop ebp
004052E8 C21000            ret $0010
```

全てのパラメータがスタックに積まれる
(順番に注意)

戻値を格納

スタックから16バイトをクリアしてリターン

Windows(x86)の呼出規約 - cdecl規約

```
CC_Cdecl.dpr.15: RetVal := Foo(1,2,3,4);
004060D4 6A04              push $04
004060D6 6A03              push $03
004060D8 6A02              push $02
004060DA 6A01              push $01
004060DC E8EFF1FFFF        call Foo
004060E1 83C410            add esp,$10
004060E4 A39CAB4000        mov [$0040ab9c],eax

CC_Cdecl.dpr.8: begin
004052D0 55                push ebp
004052D1 8BEC              mov ebp,esp
004052D3 51                push ecx
CC_Cdecl.dpr.9: Result := Param1 + Param2 + Param3 + Param4;
004052D4 8B4508            mov eax,[ebp+$08]
004052D7 03450C            add eax,[ebp+$0c]
004052DA 034510            add eax,[ebp+$10]
004052DD 034514            add eax,[ebp+$14]
004052E0 8945FC            mov [ebp-$04],eax
CC_Cdecl.dpr.10: end;
004052E3 8B45FC            mov eax,[ebp-$04]
004052E6 59                pop ecx
004052E7 5D                pop ebp
004052E8 C3                ret
```

全てのパラメータがスタックに積まれる

スタックから16バイトをクリア

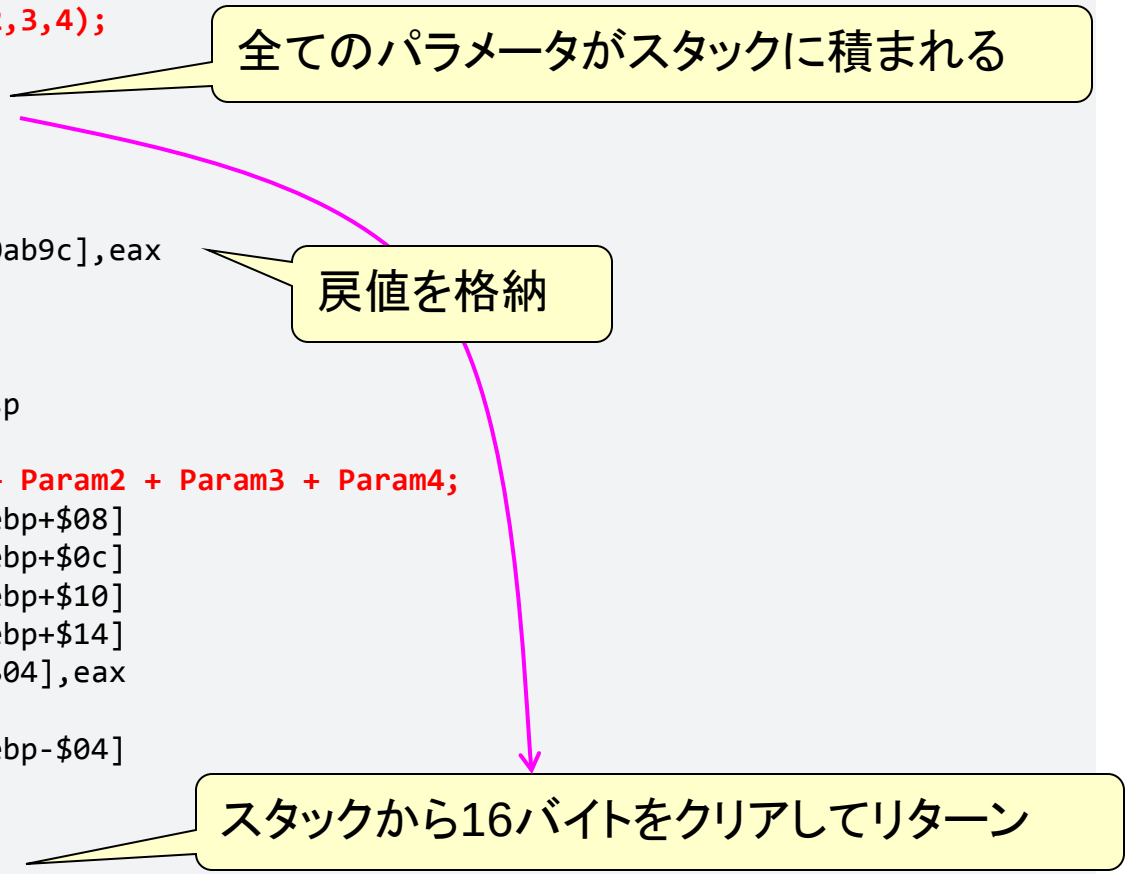
戻値を格納

単純にリターン

Windows(x86)の呼出規約 - stdcall規約

```
CC_Stdcall.dpr.15: RetVal := Foo(1,2,3,4);
004060D4 6A04              push $04
004060D6 6A03              push $03
004060D8 6A02              push $02
004060DA 6A01              push $01
004060DC E8EFF1FFFF        call Foo
004060E1 A39CAB4000        mov [$0040ab9c],eax

CC_Stdcall.dpr.8: begin
004052D0 55                push ebp
004052D1 8BEC              mov ebp,esp
004052D3 51                push ecx
CC_Stdcall.dpr.9: Result := Param1 + Param2 + Param3 + Param4;
004052D4 8B4508            mov eax,[ebp+$08]
004052D7 03450C            add eax,[ebp+$0c]
004052DA 034510            add eax,[ebp+$10]
004052DD 034514            add eax,[ebp+$14]
004052E0 8945FC            mov [ebp-$04],eax
CC_Stdcall.dpr.10: end;
004052E3 8B45FC            mov eax,[ebp-$04]
004052E6 59                pop ecx
004052E7 5D                pop ebp
004052E8 C21000            ret $0010
```



Windows(x86)の呼出規約 - safecall規約 (1)

```
CC_Safecall.dpr.15: RetVal := Foo(1,2,3,4);
004060D4 8D45EC            lea eax,[ebp-$14]
004060D7 50                push eax
004060D8 6A04              push $04
004060DA 6A03              push $03
004060DC 6A02              push $02
004060DE 6A01              push $01
004060E0 E8E3F2FFFF        call Foo
004060E5 E85EEAFFFF        call @CheckAutoResult
004060EA 8B45EC            mov eax,[ebp-$14]
004060ED A3A0AB4000        mov [$0040aba0],eax

CC_Safecall.dpr.8: begin
004053C8 55                push ebp
004053C9 8BEC              mov ebp,esp
004053CB 53                push ebx
004053CC 56                push esi
004053CD 57                push edi
004053CE 6A00              push $00
004053D0 33C0              xor eax,eax
004053D2 55                push ebp
004053D3 68FC534000        push $004053fc
004053D8 64FF30            push dword ptr fs:[eax]
004053DB 648920            mov fs:[eax],esp
```

戻値を格納するアドレス(隠しパラメータ)

全てのパラメータがスタックに積まれる

エラーがあったかどうかを検査

戻値を格納

スタック上に例外処理構造体を構築してFS:[0]に登録

Windows(x86)の呼出規約 - safecall規約 (2)

```
CC_Safecall.dpr.9: Result := Param1 + Param2 + Param3 + Param4;
004053DE 8B4508            mov eax,[ebp+$08]
004053E1 03450C            add eax,[ebp+$0c]
004053E4 034510            add eax,[ebp+$10]
004053E7 034514            add eax,[ebp+$14]
004053EA 8B5518            mov edx,[ebp+$18]
004053ED 8902              mov [edx],eax
CC_Safecall.dpr.10: end;
004053EF 33C0              xor eax,eax
004053F1 5A                pop edx
004053F2 59                pop ecx
004053F3 59                pop ecx
004053F4 648910            mov fs:[eax],edx
004053F7 58                pop eax
004053F8 33DB              xor ebx,ebx
004053FA EB05              jmp $00405401
004053FC E90FEAFFFF        jmp @HandleAutoException
00405401 8BC3              mov eax,ebx
00405403 5F                pop edi
00405404 5E                pop esi
00405405 5B                pop ebx
00405406 5D                pop ebp
00405407 C21400            ret $0014
```

戻値を隠しパラメータの場所に格納

スタック上の例外処理構造体を
クリーンアップ

エラー発生時はここから
エラーハンドラにジャンプ

スタックから20バイトをクリアしてリターン

embarcadero
Developer Camp

クラスのメソッドの扱い

クラスのメソッドの扱い(1)

- 暗黙でSelfが渡される
  - インスタンスメソッドではインスタンスへのポインタ
  - クラスメソッドではクラス参照型へのポインタ
- コンストラクタおよびデストラクタでは追加で呼出コンテキストのフラグ(8bit)が渡される
  - True(0x01)は直接の(最上位の)コンストラクタ/デストラクタ呼出を意味する(インスタンス領域の確保/解放が行われる)
  - False(0x00)は`inherited`による間接的なコンストラクタ/デストラクタ呼出を意味する(通常のメソッドと同様)

クラスのメソッドの扱い(2)

- サンプル

```
type
  TTest = class(TObject)
  public
    constructor Create;
  end;

constructor TTest.Create;
begin
  inherited;
end;

var
  Test: TTest;
begin
  Test := TTest.Create;
  Test.Free;
end.
```

クラスのメソッドの扱い(3)

- コンストラクタ(呼出元)

```
CC_OBJECT.dpr.24: Test := TTest.Create;
0041B394 B201              mov dl,$01
0041B396 A108914100        mov eax,[$00419108]
0041B39B E824DEFFFF        call TTest.Create
0041B3A0 A3D82E4200        mov [$00422ed8],eax
```

DLに0x01(True)を格納

EAXにTTestのクラス参照を格納

戻値を格納

クラスのメソッドの扱い(3)

- コンストラクタ(呼出先)

```
CC_OBJECT.dpr.17: begin
004191C4 55                push ebp
004191C5 8BEC              mov ebp,esp
004191C7 83C4F8            add esp,-$08
004191CA 84D2              test dl,dl
004191CC 7408              jz $004191d6
004191CE 83C4F0            add esp,-$10
004191D1 E83EB7FEFF        call @ClassCreate
004191D6 8855FB            mov [ebp-$05],dl
004191D9 8945FC            mov [ebp-$04],eax
CC_OBJECT.dpr.18: inherited;
004191DC 33D2              xor edx,edx
004191DE 8B45FC            mov eax,[ebp-$04]
004191E1 E872B1FEFF        call TObject.Create
CC_OBJECT.dpr.19: end;
004191E6 8B45FC            mov eax,[ebp-$04]
004191E9 807DFB00          cmp byte ptr [ebp-$05],$00
004191ED 740F              jz $004191fe
004191EF E878B7FEFF        call @AfterConstruction
004191F4 648F0500000000    pop dword ptr fs:[$00000000]
004191FB 83C40C            add esp,$0c
004191FE 8B45FC            mov eax,[ebp-$04]
00419201 59                pop ecx
00419202 59                pop ecx
00419203 5D                pop ebp
00419204 C3                ret
```

呼出コンテキストフラグ(DL)が0x00でなければClassCreateを呼び出す

DLをクリアしてから継承元のコンストラクタを呼び出す

呼出コンテキストフラグが0x00でなければAfterConstructionを呼び出す

embarcadero
Developer Camp

Windows(x64)の呼出規約

Windows(x64)の呼出規約(1)

- Windows(x64) の呼出規約は一つしかない
  - ただしsafecallは例外ファイアウォールを持つので別扱い
- 最大4つの64bit値をレジスタで渡すことができる
  - RCX/RDX/R8/R9レジスタを使用
  - 実数型はXMM0L/XMM1L/XMM2L/XMM3Lを使用
  - スタック上にもレジスタ経由の領域を確保する
- 戻値はRAXレジスタに格納される
- メモリ上のレイアウトは8バイト(QWORD)単位

Windows(x64)の呼出規約(2)

- サンプル
  - 5つのパラメータを渡してみます

スタックを使用させるため

```
function Foo(Param1: Integer; Param2: Integer; Param3: Integer; Param4: Integer;
             Param5: Integer): Integer;
begin
  Result := Param1 + Param2 + Param3 + Param4 + Param5;
end;

var
  RetVal: Integer;
begin
  RetVal := Foo(1,2,3,4,5);
  Writeln(RetVal);
end.
```

Windows(x64)の呼出規約(3)

```
CC_x64.dpr.15: RetVal := Foo(1,2,3,4,5);
000000000040B6AF C7C101000000      mov ecx,$00000001
000000000040B6B5 C7C202000000      mov edx,$00000002
000000000040B6BB 41C7C003000000    mov r8d,$00000003
000000000040B6C2 41C7C104000000    mov r9d,$00000004
000000000040B6C9 C744242005000000 mov [rsp+$20],$00000005
000000000040B6D1 E8CAF7FFFF        call Foo
000000000040B6D6 8905F4750000      mov [rel $000075f4],eax

CC_x64.dpr.8: begin
000000000040AEA0 55                push rbp
000000000040AEA1 4883EC10          sub rsp,$10
000000000040AEA5 488BEC            mov rbp,rsp
000000000040AEA8 894D20            mov [rbp+$20],ecx
000000000040AEAB 895528            mov [rbp+$28],edx
000000000040AEAE 44894530          mov [rbp+$30],r8d
000000000040AEB2 44894D38          mov [rbp+$38],r9d
CC_x64.dpr.9: Result := Param1 + Param2 + Param3 + Param4 + Param5;
000000000040AEB6 8B4520            mov eax,[rbp+$20]
000000000040AEB9 034528            add eax,[rbp+$28]
000000000040AEBC 034530            add eax,[rbp+$30]
000000000040AEBF 034538            add eax,[rbp+$38]
000000000040AEC2 034540            add eax,[rbp+$40]
000000000040AEC5 89450C            mov [rbp+$0c],eax
CC_x64.dpr.10: end;
000000000040AEC8 8B450C            mov eax,[rbp+$0c]
000000000040AECB 488D6510          lea rsp,[rbp+$10]
000000000040AECF 5D                pop rbp
000000000040AED0 C3                ret
```

第5パラメータだけがスタックに積まれ、残りはレジスタに格納

戻値を格納

パラメータをスタック上の予約領域(spill)にもコピー(コンサバティブな実装)

単純にリターン

embarcadero
Developer Camp

参考文献

参考文献 (1)

- RAD Studioのヘルプ
  - Delphi リファレンス→Delphi 言語ガイド→プログラムの制御
    - http://docwiki.embarcadero.com/RADStudio/ja/%E3%83%97%E3%83%AD%E3%82%B0%E3%83%A9%E3%83%A0%E3%81%AE%E5%88%B6%E5%BE%A1
  - Delphi リファレンス→Delphi 言語ガイド→手続きと関数→手続きと関数
    - http://docwiki.embarcadero.com/RADStudio/ja/%E6%89%8B%E7%B6%9A%E3%81%8D%E3%81%A8%E9%96%A2%E6%95%B0

参考文献 (2)

- RAD Studioのヘルプ
  - IDEリファレンス→RAD Studioのダイアログと
    コマンド→[表示]メニュー→デバッグ ウィンドウ
    →CPU ウィンドウ
    - http://docwiki.embarcadero.com/RADStudio/ja/%EF%BC%BB%E9%80%86%E3%82%A2%E3%82%BB%E3%83%B3%E3%83%96%E3%83%AB%EF%BC%BD%E3%83%9A%E3%82%A4%E3%83%B3

参考文献 (3)

- MSDN
  - 呼出規約
    - http://msdn.microsoft.com/ja-jp/library/9b372w95.aspx
  - Calling Conventions
    - http://msdn.microsoft.com/en-us/library/k2b2ssfy.aspx
  - x64 呼び出し規約の概要
    - http://msdn.microsoft.com/ja-jp/library/ms235286

参考文献 (4)

- The Old New Thing (Raymond Chenさんのblog)
  - http://blogs.msdn.com/b/oldnewthing/
  - The history of calling conventions, part 1~5
    - http://blogs.msdn.com/b/oldnewthing/archive/2004/01/02/47184.aspx
    - http://blogs.msdn.com/b/oldnewthing/archive/2004/01/07/48303.aspx
    - http://blogs.msdn.com/b/oldnewthing/archive/2004/01/08/48616.aspx
    - http://blogs.msdn.com/b/oldnewthing/archive/2004/01/13/58199.aspx
    - http://blogs.msdn.com/b/oldnewthing/archive/2004/01/14/58579.aspx

参考文献 (5)

- Intel
  - 日本語技術資料のダウンロード
    - http://www.intel.com/jp/download/index.htm
  - IA-32 インテルアーキテクチャ
    ソフトウェア・デベロッパーズ・マニュアル
    - 中巻A：命令セット・リファレンスA-M
      - http://download.intel.com/jp/developer/jpdoc/IA32_Arh_Dev_Man_Vol2A_i.pdf
    - 中巻B：命令セット・リファレンスN-Z
      - http://download.intel.com/jp/developer/jpdoc/IA32_Arh_Dev_Man_Vol2B_i.pdf

参考文献 (6)

- Intel
  - インテル エクステンデッド・メモリ 64 テクノロジ・ソフトウェア・デベロッパーズ・ガイド
    - 第1巻
      - http://download.intel.com/jp/developer/jpdoc/EM64T_VOL1_30083402_i.pdf
    - 第2巻
      - http://download.intel.com/jp/developer/jpdoc/EM64T_VOL2_30083502_i.pdf

embarcadero
Developer Camp

Q & A

Q & A

- 不明な点は公式フォーラムかDelphi-mlで
  - Embarcadero Discussion Forums: Delphi
    - https://forums.embarcadero.com/forum.jspa?forumID=14
  - Delphi | freeml
    - http://www.freeml.com/delphi-users

embarcadero
Developer Camp

ありがとうございました