DCU(Delphi Compiled Unit) Format

Header

From unofficial sources, look at 4th byte of the .dcu

```
$0F = Delphi 7
$11 = Delphi 2005
$12 = Delphi 2006 or 2007
$14 = Delphi 2009
$15 = Delphi 2010
$16 = Delphi XE
$17 = Delphi XE2
$18 = Delphi XE3
$19 = Delphi XE4
$1A = Delphi XE5
$1B = Delphi XE6
$1C = Delphi XE7
$1D = Delphi XE8
$1E = Delphi 10 Seattle
$1F = Delphi 10.1 Berlin
$20 = Delphi 10.2
$21 = Delphi 10.3
$22 = Delphi 10.4
$23 = Delphi 11
$24 = delphi 12
```

There was no change in .dcu format going from Delphi 2006 to Delphi 2007. Therefore they use the same.

Edit Jul 2, 2016 Added XE8, 10 and 10.1 to the list.

On request, also the target platform, which is found in the second byte of the .dcu. Values are of course valid only for versions that have these targets.

```
$03 = Win32
$23 = Win64
$04 = Osx32
$14 = iOS emulator
$76 = iOS device
$77 = Android
```

Some useful sites

Delphi related

1. DCU32INT: http://hmelnov.icc.ru/DCU/index.eng.html

   source: https://gitlab.com/dcu32int/DCU32INT

   The utility DCU32INT parses *.dcu file and converts it into a close to Pascal form. See DCU32INT.txt for more details. The unit versions supported are Delphi 2.0-8.0, 2005-2006/Turbo Delphi (.net and WIN32), 2007-2010 (WIN32), XE (WIN32), XE2-XE3 (WIN32,WIN64,OSX32), XE4 (WIN32,WIN64,OSX32,iOS simulator, iOS device (no code)), XE5-XE7/AppMethod (WIN32,WIN64,OSX32,iOS simulator, iOS device (no code), Android (no code)), XE8, 10 Seattle, 10.1 Berlin (WIN32,WIN64,OSX32,iOS simulator, iOS device 32/64 (no code),Android (no code)), 10.2 Tokyo (WIN32,WIN64,OSX32,iOS simulator, iOS device 32/64 (no code),Android (no code),Linux (no code)), 10.3 Rio (WIN32,WIN64,OSX32,iOS simulator, iOS device 32/64 (no code),Android (no code),Linux (may be - not checked,no code)), Kylix 1.0-3.0.

2. IDR (Interactive Delphi Reconstructor): https://github.com/crypto2011/IDR

   A decompiler of executable files (EXE) and dynamic libraries (DLL), written in Delphi and executed in Windows32 environment. Final project goal is development of the program capable to restore the most part of initial Delphi source codes from the compiled file but IDR, as well as others Delphi decompilers, cannot do it yet. Nevertheless, IDR is in a status considerably to facilitate such process. In comparison with other well known Delphi decompilers the result of IDR analysis has the greatest completeness and reliability.

3. revendepro: http://www.ggoossen.net/revendepro/

   Revendepro finds almost all structures (classes, types, procedures, etc) in the program, and generates the pascal representation, procedures will be written in assembler. Due to some limitation in assembler the generated output can not be recompiled. The source to this decompiler is freely available. Unfortunately this is the only one decompiler I was not able to use - it prompts with an exception when you try to decompile some Delphi executable file.

4. EMS Source Rescuer: https://ems-source-rescuer.apponic.com/

   EMS Source Rescuer is an easy-to-use wizard application which can help you to restore your lost source code. If you lose your Delphi or C++Builder project sources, but have an executable file, then this tool can rescue part of lost sources. Rescuer produces all project forms and data modules with all assigned properties and events. Produced event procedures don't have a body (it is not a decompiler), but have an address of code in executable file. In most cases Rescuer saves 50-90% of your time to project restoration.

5. Dede: http://www.softpedia.com/get/Programming/Debuggers-Decompilers-Dissasemblers/DeDe.shtml

   source: https://github.com/Hanvdm/dedex

   DeDe is a very fast program that can analyze executables compiled with Delphi. After decompilation DeDe gives you the following:

   - All dfm files of the target. You will be able to open and edit them with Delphi.
   - All published methods in well commented ASM code with references to strings, imported function calls, classes methods calls, components in the unit, Try-Except and Try-Finally blocks. By default DeDe retrieves only the published methods sources, but you may also process another procedure in a executable if you know the RVA offset using the Tools| Disassemble Proc menu.
   - A lot of additional information.
   - You can create a Delphi project folder with all dfm, pas, dpr files. Note: pas files contains the mentioned above well commented ASM code. They can not be recompiled!

others

https://www.agner.org/optimize/