



# CUDA BINARY UTILITIES

DA-06762-001\_v7.0 | March 2015

## Application Note



# TABLE OF CONTENTS

|   |           |
|---|-----------|
| <b>Chapter 1. Overview.....</b>                     | <b>1</b>  |
| 1.1. What is a CUDA Binary?.....                    | 1         |
| 1.2. Differences between cuobjdump and nvdiasm..... | 1         |
| <b>Chapter 2. cuobjdump.....</b>                    | <b>3</b>  |
| 2.1. Usage.....                                     | 3         |
| 2.2. Command-line Options.....                      | 5         |
| <b>Chapter 3. nvdiasm.....</b>                      | <b>7</b>  |
| 3.1. Usage.....                                     | 7         |
| 3.2. Command-line Options.....                      | 10        |
| <b>Chapter 4. Instruction Set Reference.....</b>    | <b>12</b> |
| 4.1. GT200 Instruction Set.....                     | 12        |
| 4.2. Fermi Instruction Set.....                     | 14        |
| 4.3. Kepler Instruction Set.....                    | 18        |
| 4.4. Maxwell Instruction Set.....                   | 21        |
| <b>Chapter 5. nvprune.....</b>                      | <b>26</b> |
| 5.1. Usage.....                                     | 26        |
| 5.2. Command-line Options.....                      | 26        |

## LIST OF FIGURES

|                                   |   |
|-----------------------------------|---|
| Figure 1 Control Flow Graph ..... | 9 |
|-----------------------------------|---|

## LIST OF TABLES

|         |   |    |
|---------|---|----|
| Table 1 | Comparison of cuobjdump and nvdiasm ..... | 1  |
| Table 2 | cuobjdump Command-line Options .....      | 5  |
| Table 3 | nvdiasm Command-line Options .....        | 11 |
| Table 4 | GT200 Instruction Set .....               | 12 |
| Table 5 | Fermi Instruction Set .....               | 15 |
| Table 6 | Kepler Instruction Set .....              | 18 |
| Table 7 | Maxwell Instruction Set .....             | 21 |
| Table 8 | nvprune Command-line Options .....        | 27 |

# Chapter 1.

## OVERVIEW

This document introduces **cuobjdump**, **nvdiasm**, and **nvprune**, three CUDA binary tools for Linux(x86, POWER8 and ARM), Windows, Mac OS and Android.

### 1.1. What is a CUDA Binary?

A CUDA binary (also referred to as cubin) file is an ELF-formatted file which consists of CUDA executable code sections as well as other sections containing symbols, relocators, debug info, etc. By default, the CUDA compiler driver **nvcc** embeds cubin files into the host executable file. But they can also be generated separately by using the "**-cubin**" option of **nvcc**. cubin files are loaded at run time by the CUDA driver API.



For more details on cubin files or the CUDA compilation trajectory, refer to [NVIDIA CUDA Compiler Driver NVCC](#).

### 1.2. Differences between **cuobjdump** and **nvdiasm**

CUDA provides two binary utilities for examining and disassembling cubin files and host executables: **cuobjdump** and **nvdiasm**. Basically, **cuobjdump** accepts both cubin files and host binaries while **nvdiasm** only accepts cubin files; but **nvdiasm** provides richer output options.

Here's a quick comparison of the two tools:

Table 1 Comparison of **cuobjdump** and **nvdiasm**

|                   | <b>cuobjdump</b> | <b>nvdiasm</b> |
|-------------------|------------------|----------------|
| Disassemble cubin | Yes              | Yes            |

|  | <b>cuobjdump</b> | <b>nvdiasm</b> |
|--|------------------|----------------|
| Extract ptx and extract and disassemble cubin from the following input files: <ul style="list-style-type: none"> <li>▶ Host binaries               <ul style="list-style-type: none"> <li>▶ Executables</li> <li>▶ Object files</li> <li>▶ Static libraries</li> </ul> </li> <li>▶ External fatbinary files</li> </ul> | Yes              | No             |
| Control flow analysis and output   | No               | Yes            |
| Advanced display options   | No               | Yes            |

# Chapter 2.

## CUOBJDUMP

**cuobjdump** extracts information from CUDA binary files (both standalone and those embedded in host binaries) and presents them in human readable format. The output of **cuobjdump** includes CUDA assembly code for each kernel, CUDA ELF section headers, string tables, relocators and other CUDA specific sections. It also extracts embedded ptx text from host binaries.

For a list of CUDA assembly instruction set of each GPU architecture, see [Instruction Set Reference](#).

## 2.1. Usage

**cuobjdump** accepts a single input file each time it's run. The basic usage is as following:

```
cuobjdump [options] <file>
```

To disassemble a standalone cubin or cubins embedded in a host executable and show CUDA assembly of the kernels, use the following command:

```
cuobjdump -sass <input file>
```

To dump cuda elf sections in human readable format from a cubin file, use the following command:

```
cuobjdump -elf <cubin file>
```

To extract ptx text from a host binary, use the following command:

```
cuobjdump -ptx <host binary>
```

Here's a sample output of **cuobjdump**:

```

$ cuobjdump a.out -ptx -sass
Fatbin elf code:
=====
arch = sm_20
code version = [1,7]
producer = cuda
host = linux
compile_size = 64bit
identifier = add.cu

    code for sm_20
        Function : _Z3addPiS_S
        .headerflags    @"EF_CUDA_SM20_EF_CUDA_PTX_SM(EF_CUDA_SM20)"
        /*0000*/        MOV R1, c[0x1][0x100]; /* 0x2800440400005de4 */
        /*0008*/        MOV R6, c[0x0][0x20]; /* 0x2800400080019de4 */
        /*0010*/        MOV R7, c[0x0][0x24]; /* 0x280040009001dde4 */
        /*0018*/        MOV R2, c[0x0][0x28]; /* 0x28004000a0009de4 */
        /*0020*/        MOV R3, c[0x0][0x2c]; /* 0x28004000b000dde4 */
        /*0028*/        LDU.E R0, [R6]; /* 0x8c00000000601c85 */
        /*0030*/        MOV R4, c[0x0][0x30]; /* 0x28004000c0011de4 */
        /*0038*/        LDU.E R2, [R2]; /* 0x8c00000000209c85 */
        /*0040*/        MOV R5, c[0x0][0x34]; /* 0x28004000d0015de4 */
        /*0048*/        IADD R0, R2, R0; /* 0x4800000000201c03 */
        /*0050*/        ST.E [R4], R0; /* 0x9400000000401c85 */
        /*0058*/        EXIT; /* 0x80000000000001de7 */
        .....

Fatbin ptx code:
=====
arch = sm_20
code version = [4,0]
producer = cuda
host = linux
compile_size = 64bit
compressed
identifier = add.cu

.version 4.0
.target sm_20
.address_size 64

.visible .entry _Z3addPiS_S_(
.param .u64 _Z3addPiS_S__param_0,
.param .u64 _Z3addPiS_S__param_1,
.param .u64 _Z3addPiS_S__param_2
)
{
.reg .s32 %r<4>;
.reg .s64 %rd<7>;

ld.param.u64 %rd1, [_Z3addPiS_S__param_0];
ld.param.u64 %rd2, [_Z3addPiS_S__param_1];
ld.param.u64 %rd3, [_Z3addPiS_S__param_2];
cvta.to.global.u64 %rd4, %rd3;
cvta.to.global.u64 %rd5, %rd2;
cvta.to.global.u64 %rd6, %rd1;
ldu.global.u32 %r1, [%rd6];
ldu.global.u32 %r2, [%rd5];
add.s32 %r3, %r2, %r1;
st.global.u32 [%rd4], %r3;
ret;
}

```

As shown in the output, the `a.out` host binary contains cubin and ptx code for `sm_20`.



To list cubin files in the host binary use **-l`elf`** option:

```
$ cuobjdump a.out -lelf
ELF file      1: add_new.sm_20.cubin
ELF file      2: add_new.sm_30.cubin
ELF file      3: add_old.sm_20.cubin
ELF file      4: add_old.sm_30.cubin
```

To extract all the cubins as files from the host binary use **-x`elf` all** option:

```
$ cuobjdump a.out -xelf all
Extracting ELF file      1: add_new.sm_20.cubin
Extracting ELF file      2: add_new.sm_30.cubin
Extracting ELF file      3: add_old.sm_20.cubin
Extracting ELF file      4: add_old.sm_30.cubin
```

To extract the cubin named **add\_new.sm\_30.cubin**:

```
$ cuobjdump a.out -xelf add_new.sm_30.cubin
Extracting ELF file      1: add_old.sm_20.cubin
```

To extract only the cubins containing **\_old** in their names:

```
$ cuobjdump a.out -xelf _old
Extracting ELF file      1: add_old.sm_20.cubin
Extracting ELF file      2: add_old.sm_30.cubin
```

You can pass any substring to **-x`elf`** and **-x`ptx`** options. Only the files having the substring in the name will be extracted from the input binary.

## 2.2. Command-line Options

Table 2 contains supported command-line options of **cuobjdump**, along with a description of what each option does. Each option has a long name and a short name, which can be used interchangeably.

Table 2 **cuobjdump** Command-line Options

| Option (long)             | Option (short)  | Description   |
|---------------------------|-----------------|---|
| <b>--all-fatbin</b>       | <b>-all</b>     | Dump all fatbin sections. By default will only dump contents of executable fatbin (if exists), else relocatable fatbin if no executable fatbin. |
| <b>--dump-elf</b>         | <b>-elf</b>     | Dump ELF Object sections.   |
| <b>--dump-elf-symbols</b> | <b>-symbols</b> | Dump ELF symbol names.  |
| <b>--dump-ptx</b>         | <b>-ptx</b>     | Dump PTX for all listed device functions.   |

| Option (long)   | Option (short)     | Description  |
|---|--------------------|--|
| <code>--dump-sass</code>                                      | <code>-sass</code> | Dump CUDA assembly for a single cubin file or all cubin files embedded in the binary.  |
| <code>--extract-elf &lt;partial file name&gt;,...</code>      | <code>-xelf</code> | Extract ELF file(s) name containing <partial file name> and save as file(s). Use 'all' to extract all files. To get the list of ELF files use -lelf option. Works with host executable/object/library and external fatbin. All 'dump' and 'list' options are ignored with this option. |
| <code>--extract-ptx &lt;partial file name&gt;,...</code>      | <code>-xptx</code> | Extract PTX file(s) name containing <partial file name> and save as file(s). Use 'all' to extract all files. To get the list of PTX files use -lptx option. Works with host executable/object/library and external fatbin. All 'dump' and 'list' options are ignored with this option. |
| <code>--function &lt;function name&gt;,...</code>             | <code>-fun</code>  | Specify names of device functions whose fat binary structures must be dumped.  |
| <code>--gpu-architecture &lt;gpu architecture name&gt;</code> | <code>-arch</code> | Specify GPU Architecture for which information should be dumped. Allowed values for this option: 'sm_20','sm_21','sm_30','sm_32','sm_35','sm_37','sm_50','sm_52' .   |
| <code>--help</code>   | <code>-h</code>    | Print this help information on this tool.  |
| <code>--list-elf</code>                                       | <code>-lelf</code> | List all the ELF files available in the fatbin. Works with host executable/object/library and external fatbin. All other options are ignored with this flag. This can be used to select particular ELF with -xelf option later.  |
| <code>--list-ptx</code>                                       | <code>-lptx</code> | List all the PTX files available in the fatbin. Works with host executable/object/library and external fatbin. All other options are ignored with this flag. This can be used to select particular PTX with -xptx option later.  |
| <code>--options-file &lt;file&gt;,...</code>                  | <code>-optf</code> | Include command line options from specified file.  |
| <code>--sort-functions</code>                                 | <code>-sort</code> | Sort functions when dumping sass.  |
| <code>--version</code>  | <code>-v</code>    | Print version information on this tool.  |

# Chapter 3.

## NVDISASM

**nvdiasm** extracts information from standalone cubin files and presents them in human readable format. The output of **nvdiasm** includes CUDA assembly code for each kernel, listing of ELF data sections and other CUDA specific sections. Output style and options are controlled through **nvdiasm** command-line options. **nvdiasm** also does control flow analysis to annotate jump/branch targets and makes the output easier to read.



**nvdiasm** requires complete relocation information to do control flow analysis. If this information is missing from the CUDA binary, either use the **nvdiasm** option **"-ndf"** to turn off control flow analysis, or use the **ptxas** and **nvlink** option **"-preserve-relocs"** to re-generate the cubin file.

For a list of CUDA assembly instruction set of each GPU architecture, see [Instruction Set Reference](#).

### 3.1. Usage

**nvdiasm** accepts a single input file each time it's run. The basic usage is as following:

```
nvdiasm [options] <input cubin file>
```

To get the control flow graph of a kernel, use the following:

```
nvdiasm -cfg <input cubin file>
```

Here's a sample output of **nvdiasm**:

```

        .headerflags    @"EF_CUDA_TEXMODE_UNIFIED EF_CUDA_64BIT_ADDRESS
EF_CUDA_SM30 EF_CUDA_PTX_SM(EF_CUDA_SM30) "

//----- .nv.info -----
        .section        .nv.info,"",@"SHT_CUDA_INFO "
        .align 4

.....

//----- .text._Z4addXPii -----
        .section        .text._Z4addXPii,"ax",@progbits
        .sectioninfo    @"SHI_REGISTERS=11 "
        .align 4
        .global         _Z4addXPii
        .type           _Z4addXPii,@function
        .size           _Z4addXPii,(.L_19 - _Z4addXPii)
        .other          _Z4addXPii,@"STO_CUDA_ENTRY STV_DEFAULT "
_Z4addXPii:
.text._Z4addXPii:
        /*0008*/        MOV R1, c[0x0][0x44];
        /*0010*/        ISUB R1, R1, 0x8;
        /*0018*/        MOV R0, c[0x0][0x148];
        /*0020*/        IADD R6.CC, R1, c[0x0][0x24];
        /*0028*/        ISETP.LT.AND P0, PT, R0, 0x1, PT;
        /*0030*/        MOV R8, c[0x0][0x140];
        /*0038*/        MOV R9, c[0x0][0x144];
        /*0048*/        IADD.X R7, RZ, RZ;
        /*0050*/        ISUB R10, R6, c[0x0][0x24];
        /*0058*/        @P0 BRA `(.L_2);
        /*0060*/        LD.E R0, [R8];
        /*0068*/        MOV R2, RZ;
        /*0070*/        NOP;
        /*0078*/        NOP;

.L_3:
        /*0088*/        IADD R2, R2, 0x1;
        /*0090*/        MOV R3, R0;
        /*0098*/        IADD R0, R0, 0x1;
        /*00a0*/        ISETP.LT.AND P0, PT, R2, c[0x0][0x148], PT;
        /*00a8*/        @P0 BRA `(.L_3);
        /*00b0*/        IADD R0, R3, 0x1;
        /*00b8*/        ST.E [R8], R0;

.L_2:
        /*00c8*/        S2R R0, SR_TID.X;
        /*00d0*/        ISETP.NE.AND P0, PT, R0, RZ, PT;
        /*00d8*/        @P0 EXIT ;
        /*00e0*/        LD.E R0, [R8];
        /*00e8*/        MOV R4, c[0x0][0xf0];
        /*00f0*/        MOV R5, c[0x0][0xf4];
        /*00f8*/        STL [R10], R0;
        /*0108*/        JCAL `(vprintf);
        /*0110*/        EXIT ;

.L_4:
        /*0118*/        BRA `(.L_4);

.L_19:

//----- SYMBOLS -----

        .type           vprintf,@function

```

**nvdiasm** is capable of generating control flow of CUDA assembly in the format of DOT graph description language. The output of the control flow from **nvdiasm** can be directly imported to a DOT graph visualization tool such as [Graphviz](#).



This feature is only supported on cubins generated for Compute Capability 3.0 and later.

Here's how you can generate a PNG image (**cfg.png**) of the control flow of the above cubin (**a.cubin**) with **nvdiasm** and Graphviz:

```
nvdiasm -cfg a.cubin | dot -ocfg.png -Tpng
```

Here's the generated graph:

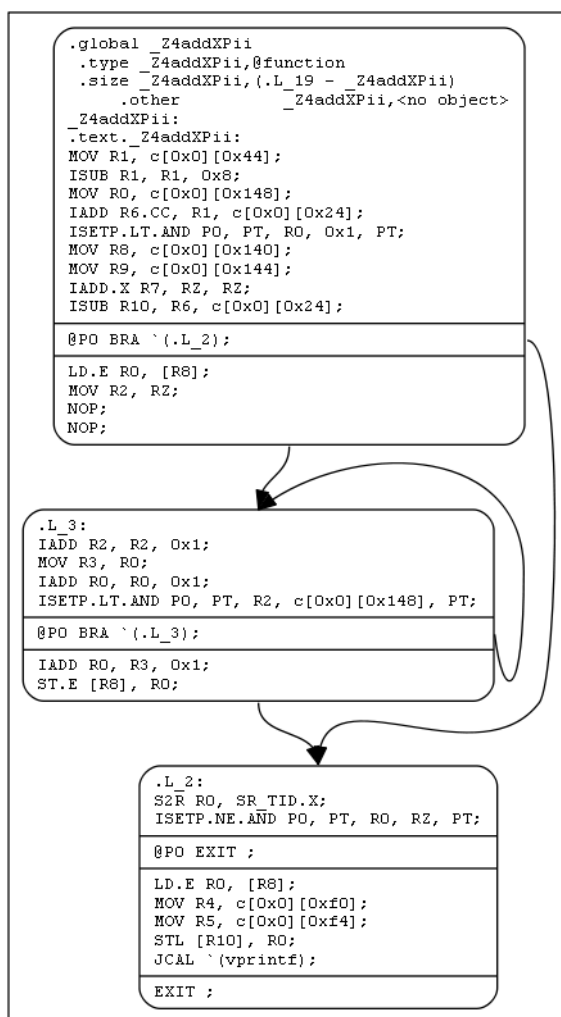
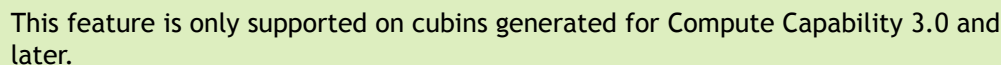


Figure 1 Control Flow Graph

**nvdiasm** is capable of showing the register (CC, general and predicate) liveness range information. For each line of CUDA assembly, **nvdiasm** displays whether a given device register was assigned, accessed, live or re-assigned. It also shows the total



```

_main10acosParams
_main10acosParams,@function
_main10acosParams, (.L_17 - _Z9acos_main10acosParams)
_main10acosParams, @.STO_CUDA_ENTRY STV_DEFAULT"

MOV R1, c[0x0][0x44];
S2R R0, SR_CTAID.X;
S2R R3, SR_TID.X;
IMAD R3, R0, c[0x0][0x28], R3;
MOV R0, c[0x0][0x28];
ISETP.GE.AND P0, PT, R3, c[0x0][0x150], PT;
IMUL R0, R0, c[0x0][0x34];
@P0 EXIT;
MOV32I R8, 0x4;
MOV32I R9, 0x3c94d2e9;
NOP;
NOP;
NOP;
NOP;

IMAD R6.CC, R3, R8, c[0x0][0x140];
IMAD.HI.X R7, R3, R8, c[0x0][0x144];
LD.E R2, [R6];
FADD.FTZ R4, -[R2], 1;
FSETP.GT.FTZ.AND P0, PT, [R2], c[0x2][0x0], PT;
FMUL.FTZ R4, R4, 0.5;
F2F.FTZ.F32.F32 R5, [R2];
MUFU.RSQ R4, R4;
@P0 MUFU.RCP R5, R4;
FMUL.FTZ R4, R5, R5;
IMAD R6.CC, R3, R8, c[0x0][0x148];
FFMA.FTZ R7, R4, c[0x2][0x4], R9;
FFMA.FTZ R7, R7, R4, c[0x2][0x8];
FFMA.FTZ R7, R7, R4, c[0x2][0xc];
FFMA.FTZ R7, R7, R4, c[0x2][0x10];
FMUL.FTZ R4, R7, R4;
IMAD.HI.X R7, R3, R8, c[0x0][0x14c];
FFMA.FTZ R4, R4, R5, R5;
IADD R3, R3, R0;
FADD32I.FTZ R5, -R4, 1.5707963705062866211;
@P0 FADD.FTZ R5, R4, R4;
ISETP.LT.AND P0, PT, R3, c[0x0][0x150], PT;
FADD32I.FTZ R4, -R5, 3.1415927410125732422;
FCMP.LT.FTZ R2, R4, R5, R2;
ST.E [R6], R2;
@P0 BRA `(.L_1);
MOV RZ, RZ;
EXIT;

BRA `(.L_2);

```

Table 3 contains the supported command-line options of **nvdiasm**, along with a description of what each option does. Each option has a long name and a short name, which can be used interchangeably.

Table 3 nvdisasm Command-line Options

| Option (long)                                | Option (short)     | Description   |
|--|--------------------|---|
| <code>--base-address &lt;value&gt;</code>    | <code>-base</code> | Specify the logical base address of the image to disassemble. This option is only valid when disassembling a raw instruction binary (see option ' <code>--binary</code> '), and is ignored when disassembling an Elf file. Default value: 0.  |
| <code>--binary &lt;SMxy&gt;</code>           | <code>-b</code>    | When this option is specified, the input file is assumed to contain a raw instruction binary, that is, a sequence of binary instruction encodings as they occur in instruction memory. The value of this option must be the asserted architecture of the raw binary. Allowed values for this option: 'SM20','SM21','SM30','SM32','SM35','SM37','SM50','SM52'.   |
| <code>--help</code>                          | <code>-h</code>    | Print this help information on this tool.   |
| <code>--life-range-mode</code>               | <code>-lrm</code>  | This option implies option ' <code>--print-life-ranges</code> ', and determines how register live range info should be printed. 'count': Not at all, leaving only the # column (number of live registers); 'wide': Columns spaced out for readability (default); 'narrow': A one-character column for each register, economizing on table width. Allowed values for this option: 'count','narrow','wide'. |
| <code>--no-dataflow</code>                   | <code>-ndf</code>  | Disable dataflow analyzer after disassembly. Dataflow analysis is normally enabled to perform branch stack analysis and annotate all instructions that jump via the GPU branch stack with inferred branch target labels. However, it may occasionally fail when certain restrictions on the input nvelv/cubin are not met.  |
| <code>--options-file &lt;file&gt;,...</code> | <code>-optf</code> | Include command line options from specified file.   |
| <code>--output-control-flow-graph</code>     | <code>-cfg</code>  | When specified, output the control flow graph in a format consumable by graphviz tools (such as dot).   |
| <code>--print-code</code>                    | <code>-c</code>    | Only print code sections.   |
| <code>--print-instruction-encoding</code>    | <code>-hex</code>  | When specified, print the encoding bytes after each disassembled operation.   |
| <code>--print-life-ranges</code>             | <code>-plr</code>  | Print register life range information in a trailing column in the produced disassembly.   |
| <code>--print-line-info</code>               | <code>-g</code>    | Annotate disassembly with source line information obtained from .debug_line section, if present.  |
| <code>--print-raw</code>                     | <code>-raw</code>  | Print the disassembly without any attempt to beautify it.   |
| <code>--separate-functions</code>            | <code>-sf</code>   | Separate the code corresponding with function symbols by some new lines to let them stand out in the printed disassembly.   |
| <code>--version</code>                       | <code>-v</code>    | Print version information on this tool.   |

# Chapter 4.

## INSTRUCTION SET REFERENCE

This is an instruction set reference of NVIDIA® GPU architectures GT200, Fermi, Kepler and Maxwell.

### 4.1. GT200 Instruction Set

The GT200 architecture (Compute Capability 1.x) has the following instruction set format:

```
(instruction) (destination) (source1), (source2) ...
```

Valid destination and source locations include:

- ▶ RX for registers
- ▶ AX for address registers
- ▶ SRX for special system-controlled registers
- ▶ CX for condition registers
- ▶ global14 r[X] for global memory referenced by an address in a register
- ▶ g[X] for shared memory
- ▶ c[X][Y] for constant memory
- ▶ local[X] for local memory

Table 4 lists valid instructions for the GT200 GPUs.

Table 4 GT200 Instruction Set

| Opcode | Description  |
|--------|--|
| A2R    | Move address register to data register                 |
| ADA    | Add immediate to address register                      |
| BAR    | CTA-wide barrier synchronization                       |
| BRA    | Conditional branch                                     |
| BRK    | Conditional break from a loop                          |
| BRX    | Fetch an address from constant memory and branch to it |



| Opcode  | Description   |
|---|---|
| C2R   | Conditional code to data register   |
| CAL   | Unconditional subroutine call   |
| COS   | Cosine  |
| DADD  | Double-precision floating point addition  |
| DFMA  | Double-precision floating point fused multiply-add  |
| DMAX  | Double-precision floating point maximum   |
| DMIN  | Double-precision floating point minimum   |
| DMUL  | Double-precision floating point multiply  |
| DSET  | Double-precision floating point conditional set   |
| EX2   | Exponential base two function   |
| F2F   | Copy floating-point value with conversion to a different floating-point type  |
| F2I   | Copy floating-point value with conversion to integer  |
| FADD/FADD32/FADD32I                                 | Single-precision floating point addition  |
| FCMP  | Single-precision floating point compare   |
| FMAD/FMAD32/FMAD32I                                 | Single-precision floating point multiply-add  |
| FMAX  | Single-precision floating point maximum   |
| FMIN  | Single-precision floating point minimum   |
| FMUL/FMUL32/FMUL32I                                 | Single-precision floating point multiply  |
| FSET  | Single-precision floating point conditional set   |
| G2R   | Move from shared memory to register. A .LCK suffix indicates that the bank is locked until a R2G.UNL has been performed; this is used to implement shared memory atomics. |
| GATOM.IADD/EXCH/CAS/IMIN/IMAX/INC/DEC/IAND/IOR/IXOR | Global memory atomic operations; performs both an atomic operation and returns the original value   |
| GLD   | Load from global memory   |
| GRED.IADD/IMIN/IMAX/INC/DEC/IAND/IOR/IXOR           | Global memory reduction operations; performs only an atomic operation with no return value  |
| GST   | Store to global memory  |
| I2F   | Copy integer value to floating-point with conversion  |
| I2I   | Copy integer value to integer with conversion   |
| IADD/IADD32/IADD32I                                 | Integer addition  |
| IMAD/IMAD32/IMAD32I                                 | Integer multiply-add  |
| IMAX  | Integer maximum   |
| IMIN  | Integer minimum   |
| IMUL/IMUL32/IMUL32I                                 | Integer multiply  |
| ISAD/ISAD32   | Sum of absolute difference  |

| Opcode    | Description  |
|-----------|--|
| ISSET     | Integer conditional set  |
| LG2       | Floating point logarithm base 2  |
| LLD       | Load from local memory   |
| LST       | Store to local memory  |
| LOP       | Logical operation (AND/OR/XOR)   |
| MOV/MOV32 | Move source to destination   |
| MVC       | Move from constant memory to destination   |
| MVI       | Move immediate to destination  |
| NOP       | No operation   |
| R2A       | Move register to address register  |
| R2C       | Move data register to conditional code   |
| R2G       | Store to shared memory. When used with the .UNL suffix, releases a previously held lock on that shared memory bank |
| RCP       | Single-precision floating point reciprocal   |
| RET       | Conditional return from subroutine   |
| RRO       | Range reduction operator   |
| RSQ       | Reciprocal square root   |
| S2R       | Move special register to register  |
| SHL       | Shift left   |
| SHR       | Shift right  |
| SIN       | Sine   |
| SSY       | Set synchronization point; used before potentially divergent instructions  |
| TEX/TEX32 | Texture fetch  |
| VOTE      | Warp-vote primitive  |

## 4.2. Fermi Instruction Set

The Fermi architecture (Compute Capability 2.x) has the following instruction set format:

```
(instruction) (destination) (source1), (source2) ...
```

Valid destination and source locations include:

- ▶ RX for registers
- ▶ SRX for special system-controlled register
- ▶ PX for condition register
- ▶ c[X][Y] for constant memory

Table 5 lists valid instructions for the Fermi GPUs.

Table 5 Fermi Instruction Set

| Opcode                             | Description                 |
|------------------------------------|-----------------------------|
| <b>Floating Point Instructions</b> |                             |
| FFMA                               | FP32 Fused Multiply Add     |
| FADD                               | FP32 Add                    |
| FCMP                               | FP32 Compare                |
| FMUL                               | FP32 Multiply               |
| FMNMX                              | FP32 Minimum/Maximum        |
| FSWZ                               | FP32 Swizzle                |
| FSET                               | FP32 Set                    |
| FSETP                              | FP32 Set Predicate          |
| RRO                                | FP Range Reduction Operator |
| MUFU                               | FP Multi-Function Operator  |
| DFMA                               | FP64 Fused Multiply Add     |
| DADD                               | FP64 Add                    |
| DMUL                               | FP64 Multiply               |
| DMNMX                              | FP64 Minimum/Maximum        |
| DSET                               | FP64 Set                    |
| DSETP                              | FP64 Set Predicate          |
| <b>Integer Instructions</b>        |                             |
| IMAD                               | Integer Multiply Add        |
| IMUL                               | Integer Multiply            |
| IADD                               | Integer Add                 |
| ISCADD                             | Integer Scaled Add          |
| ISAD                               | Integer Sum Of Abs Diff     |
| IMNMX                              | Integer Minimum/Maximum     |
| BFE                                | Integer Bit Field Extract   |
| BFI                                | Integer Bit Field Insert    |
| SHR                                | Integer Shift Right         |
| SHL                                | Integer Shift Left          |
| LOP                                | Integer Logic Op            |
| FLO                                | Integer Find Leading One    |
| ISSET                              | Integer Set                 |
| ISSETP                             | Integer Set Predicate       |

| Opcode                                 | Description   |
|--|---|
| ICMP                                   | Integer Compare and Select  |
| POPC                                   | Population Count  |
| <b>Conversion Instructions</b>         |   |
| F2F                                    | Float to Float  |
| F2I                                    | Float to Integer  |
| I2F                                    | Integer to Float  |
| I2I                                    | Integer to Integer  |
| <b>Movement Instructions</b>           |   |
| MOV                                    | Move  |
| SEL                                    | Conditional Select/Move   |
| PRMT                                   | Permute   |
| <b>Predicate/CC Instructions</b>       |   |
| P2R                                    | Predicate to Register   |
| R2P                                    | Register to Predicate   |
| CSET                                   | CC Set  |
| CSETP                                  | CC Set Predicate  |
| PSET                                   | Predicate Set   |
| PSETP                                  | Predicate Set Predicate   |
| <b>Texture Instructions</b>            |   |
| TEX                                    | Texture Fetch   |
| TLD                                    | Texture Load  |
| TLD4                                   | Texture Load 4 Texels   |
| TXQ                                    | Texture Query   |
| <b>Compute Load/Store Instructions</b> |   |
| LDC                                    | Load from Constant  |
| LD                                     | Load from Memory  |
| LDU                                    | Load Uniform  |
| LDL                                    | Load from Local Memory  |
| LDS                                    | Load from Shared Memory   |
| LDLK                                   | Load and Lock   |
| LDSLK                                  | Load from Shared Memory and Lock  |
| LD_LDU                                 | LD_LDU is a combination of a generic load LD with a load uniform LDU        |
| LDS_LDU                                | LDS_LDU is combination of a Shared window load LDS with a load uniform LDU. |
| ST                                     | Store to Memory   |

| Opcode                             | Description                        |
|------------------------------------|------------------------------------|
| STL                                | Store to Local Memory              |
| STUL                               | Store and Unlock                   |
| STS                                | Store to Shared Memory             |
| STSUL                              | Store to Shared Memory and Unlock  |
| ATOM                               | Atomic Memory Operation            |
| RED                                | Atomic Memory Reduction Operation  |
| CCTL                               | Cache Control                      |
| CCTLL                              | Cache Control (Local)              |
| MEMBAR                             | Memory Barrier                     |
| <b>Surface Memory Instructions</b> |                                    |
| SULD                               | Surface Load                       |
| SULEA                              | Surface Load Effective Address     |
| SUST                               | Surface Store                      |
| SURED                              | Surface Reduction                  |
| SUQ                                | Surface Query                      |
| <b>Control Instructions</b>        |                                    |
| BRA                                | Branch to Relative Address         |
| BRX                                | Branch to Relative Indexed Address |
| JMP                                | Jump to Absolute Address           |
| JMX                                | Jump to Absolute Indexed Address   |
| CAL                                | Call to Relative Address           |
| JCAL                               | Call to Absolute Address           |
| RET                                | Return from Call                   |
| BRK                                | Break from Loop                    |
| CONT                               | Continue in Loop                   |
| LONGJMP                            | Long Jump                          |
| SSY                                | Set Sync Relative Address          |
| PBK                                | Pre-Break Relative Address         |
| PCNT                               | Pre-Continue Relative Address      |
| PRET                               | Pre-Return Relative Address        |
| PLONGJMP                           | Pre-Long-Jump Relative Address     |
| BPT                                | Breakpoint/Trap                    |
| EXIT                               | Exit Program                       |
| <b>Miscellaneous Instructions</b>  |                                    |
| NOP                                | No Operation                       |

| Opcode | Description                    |
|--------|--------------------------------|
| S2R    | Special Register to Register   |
| B2R    | Barrier to Register            |
| LEPC   | Load Effective PC              |
| BAR    | Barrier Synchronization        |
| VOTE   | Query condition across threads |

## 4.3. Kepler Instruction Set

The Kepler architecture (Compute Capability 3.x) has the following instruction set format:

```
(instruction) (destination) (source1), (source2) ...
```

Valid destination and source locations include:

- ▶ X for registers
- ▶ SRX for special system-controlled registers
- ▶ PX for condition registers
- ▶ c[X][Y] for constant memory

Table 6 lists valid instructions for the Kepler GPUs.

Table 6 Kepler Instruction Set

| Opcode                             | Description                 |
|------------------------------------|-----------------------------|
| <b>Floating Point Instructions</b> |                             |
| FFMA                               | FP32 Fused Multiply Add     |
| FADD                               | FP32 Add                    |
| FCMP                               | FP32 Compare                |
| FMUL                               | FP32 Multiply               |
| FMNMX                              | FP32 Minimum/Maximum        |
| FSWZ                               | FP32 Swizzle                |
| FSET                               | FP32 Set                    |
| FSETP                              | FP32 Set Predicate          |
| FCHK                               | FP32 Division Test          |
| RRO                                | FP Range Reduction Operator |
| MUFU                               | FP Multi-Function Operator  |
| DFMA                               | FP64 Fused Multiply Add     |
| DADD                               | FP64 Add                    |
| DMUL                               | FP64 Multiply               |

| Opcode                           | Description                  |
|----------------------------------|------------------------------|
| DMNMX                            | FP64 Minimum/Maximum         |
| DSET                             | FP64 Set                     |
| DSETP                            | FP64 Set Predicate           |
| <b>Integer Instructions</b>      |                              |
| IMAD                             | Integer Multiply Add         |
| IMADSP                           | Integer Extract Multiply Add |
| IMUL                             | Integer Multiply             |
| IADD                             | Integer Add                  |
| ISCADD                           | Integer Scaled Add           |
| ISAD                             | Integer Sum Of Abs Diff      |
| IMNMX                            | Integer Minimum/Maximum      |
| BFE                              | Integer Bit Field Extract    |
| BFI                              | Integer Bit Field Insert     |
| SHR                              | Integer Shift Right          |
| SHL                              | Integer Shift Left           |
| SHF                              | Integer Funnel Shift         |
| LOP                              | Integer Logic Op             |
| FLO                              | Integer Find Leading One     |
| ISSET                            | Integer Set                  |
| ISETP                            | Integer Set Predicate        |
| ICMP                             | Integer Compare and Select   |
| POPC                             | Population Count             |
| <b>Conversion Instructions</b>   |                              |
| F2F                              | Float to Float               |
| F2I                              | Float to Integer             |
| I2F                              | Integer to Float             |
| I2I                              | Integer to Integer           |
| <b>Movement Instructions</b>     |                              |
| MOV                              | Move                         |
| SEL                              | Conditional Select/Move      |
| PRMT                             | Permute                      |
| SHFL                             | Warp Shuffle                 |
| <b>Predicate/CC Instructions</b> |                              |
| P2R                              | Predicate to Register        |
| R2P                              | Register to Predicate        |

| Opcode                                 | Description                                     |
|--|---|
| CSET                                   | CC Set  |
| CSETP                                  | CC Set Predicate                                |
| PSET                                   | Predicate Set                                   |
| PSETP                                  | Predicate Set Predicate                         |
| <b>Texture Instructions</b>            |   |
| TEX                                    | Texture Fetch                                   |
| TLD                                    | Texture Load                                    |
| TLD4                                   | Texture Load 4 Texels                           |
| TXQ                                    | Texture Query                                   |
| <b>Compute Load/Store Instructions</b> |   |
| LDC                                    | Load from Constant                              |
| LD                                     | Load from Memory                                |
| LDG                                    | Non-coherent Global Memory Load                 |
| LDL                                    | Load from Local Memory                          |
| LDS                                    | Load from Shared Memory                         |
| LDSLK                                  | Load from Shared Memory and Lock                |
| ST                                     | Store to Memory                                 |
| STL                                    | Store to Local Memory                           |
| STS                                    | Store to Shared Memory                          |
| STSCUL                                 | Store to Shared Memory Conditionally and Unlock |
| ATOM                                   | Atomic Memory Operation                         |
| RED                                    | Atomic Memory Reduction Operation               |
| CCTL                                   | Cache Control                                   |
| CCTLL                                  | Cache Control (Local)                           |
| MEMBAR                                 | Memory Barrier                                  |
| <b>Surface Memory Instructions</b>     |   |
| SUCLAMP                                | Surface Clamp                                   |
| SUBFM                                  | Surface Bit Field Merge                         |
| SUEAU                                  | Surface Effective Address                       |
| SULDGA                                 | Surface Load Generic Address                    |
| SUSTGA                                 | Surface Store Generic Address                   |
| <b>Control Instructions</b>            |   |
| BRA                                    | Branch to Relative Address                      |
| BRX                                    | Branch to Relative Indexed Address              |
| JMP                                    | Jump to Absolute Address                        |



| Opcode                            | Description                      |
|-----------------------------------|----------------------------------|
| JMX                               | Jump to Absolute Indexed Address |
| CAL                               | Call to Relative Address         |
| JCAL                              | Call to Absolute Address         |
| RET                               | Return from Call                 |
| BRK                               | Break from Loop                  |
| CONT                              | Continue in Loop                 |
| SSY                               | Set Sync Relative Address        |
| PBK                               | Pre-Break Relative Address       |
| PCNT                              | Pre-Continue Relative Address    |
| PRET                              | Pre-Return Relative Address      |
| BPT                               | Breakpoint/Trap                  |
| EXIT                              | Exit Program                     |
| <b>Miscellaneous Instructions</b> |                                  |
| NOP                               | No Operation                     |
| S2R                               | Special Register to Register     |
| B2R                               | Barrier to Register              |
| BAR                               | Barrier Synchronization          |
| VOTE                              | Query condition across threads   |

## 4.4. Maxwell Instruction Set

The Maxwell architecture (Compute Capability 5.x) has the following instruction set format:

```
(instruction) (destination) (source1), (source2) ...
```

Valid destination and source locations include:

- ▶ X for registers
- ▶ SRX for special system-controlled registers
- ▶ PX for condition registers
- ▶ c[X][Y] for constant memory

Table 7 lists valid instructions for the Maxwell GPUs.

**Table 7 Maxwell Instruction Set**

| Opcode                             | Description |
|------------------------------------|-------------|
| <b>Floating Point Instructions</b> |             |
| FADD                               | FP32 Add    |

| Opcode                      | Description                               |
|-----------------------------|---|
| FCHK                        | Single Precision FP Divide Range Check    |
| FCMP                        | FP32 Compare to Zero and Select Source    |
| FFMA                        | FP32 Fused Multiply and Add               |
| FMNMX                       | FP32 Minimum/Maximum                      |
| FMUL                        | FP32 Multiply                             |
| FSET                        | FP32 Compare And Set                      |
| FSETP                       | FP32 Compare And Set Predicate            |
| FSWZADD                     | FP32 Add used for FSWZ emulation          |
| MUFU                        | Multi Function Operation                  |
| RRO                         | Range Reduction Operator FP               |
| DADD                        | FP64 Add                                  |
| DFMA                        | FP64 Fused Mutiply Add                    |
| DMNMX                       | FP64 Minimum/Maximum                      |
| DMUL                        | FP64 Multiply                             |
| DSET                        | FP64 Compare And Set                      |
| DSETP                       | FP64 Compare And Set Predicate            |
| <b>Integer Instructions</b> |   |
| BFE                         | Bit Field Extract                         |
| BFI                         | Bit Field Insert                          |
| FLO                         | Find Leading One                          |
| IADD                        | Integer Addition                          |
| IADD3                       | 3-input Integer Addition                  |
| ICMP                        | Integer Compare to Zero and Select Source |
| IMAD                        | Integer Multiply And Add                  |
| IMADSP                      | Extracted Integer Multiply And Add.       |
| IMNMX                       | Integer Minimum/Maximum                   |
| IMUL                        | Integer Multiply                          |
| ISCADD                      | Scaled Integer Addition                   |
| ISSET                       | Integer Compare And Set                   |
| ISETP                       | Integer Compare And Set Predicate         |
| LEA                         | Compute Effective Address                 |
| LOP                         | Logic Operation                           |
| LOP3                        | 3-input Logic Operation                   |
| POPC                        | Population count                          |
| SHF                         | Funnel Shift                              |

| Opcode                                 | Description   |
|--|---|
| SHL                                    | Shift Left  |
| SHR                                    | Shift Right   |
| XMAD                                   | Integer Short Multiply Add                              |
| <b>Conversion Instructions</b>         |   |
| F2F                                    | Floating Point To Floating Point Conversion             |
| F2I                                    | Floating Point To Integer Conversion                    |
| I2F                                    | Integer To Floating Point Conversion                    |
| I2I                                    | Integer To Integer Conversion                           |
| <b>Movement Instructions</b>           |   |
| MOV                                    | Move  |
| PRMT                                   | Permute Register Pair                                   |
| SEL                                    | Select Source with Predicate                            |
| SHFL                                   | Warp Wide Register Shuffle                              |
| <b>Predicate/CC Instructions</b>       |   |
| CSET                                   | Test Condition Code And Set                             |
| CSETP                                  | Test Condition Code and Set Predicate                   |
| PSET                                   | Combine Predicates and Set                              |
| PSETP                                  | Combine Predicates and Set Predicate                    |
| P2R                                    | Move Predicate Register To Register                     |
| R2P                                    | Move Register To Predicate/CC Register                  |
| <b>Texture Instructions</b>            |   |
| TEX                                    | Texture Fetch   |
| TLD                                    | Texture Load  |
| TLD4                                   | Texture Load 4  |
| TXQ                                    | Texture Query   |
| TEXS                                   | Texture Fetch with scalar/non-vec4 source/destinations  |
| TLD4S                                  | Texture Load 4 with scalar/non-vec4 source/destinations |
| TLDS                                   | Texture Load with scalar/non-vec4 source/destinations   |
| <b>Compute Load/Store Instructions</b> |   |
| LD                                     | Load from generic Memory                                |
| LDC                                    | Load Constant   |
| LDG                                    | Load from Global Memory                                 |
| LDL                                    | Load within Local Memory Window                         |
| LDS                                    | Local within Shared Memory Window                       |
| ST                                     | Store to generic Memory                                 |

| Opcode                             | Description                               |
|------------------------------------|---|
| STG                                | Store to global Memory                    |
| STL                                | Store within Local or Shared Window       |
| STS                                | Store within Local or Shared Window       |
| ATOM                               | Atomic Operation on generic Memory        |
| ATOMS                              | Atomic Operation on Shared Memory         |
| RED                                | Reduction Operation on generic Memory     |
| CCTL                               | Cache Control                             |
| CCTL                               | Cache Control                             |
| MEMBAR                             | Memory Barrier                            |
| CCTLT                              | Texture Cache Control                     |
| <b>Surface Memory Instructions</b> |   |
| SUATOM                             | Surface Reduction                         |
| SULD                               | Surface Load                              |
| SURED                              | Atomic Reduction on surface memory        |
| SUST                               | Surface Store                             |
| <b>Control Instructions</b>        |   |
| BRA                                | Relative Branch                           |
| BRX                                | Relative Branch Indirect                  |
| JMP                                | Absolute Jump                             |
| JMX                                | Absolute Jump Indirect                    |
| SSY                                | Set Synchronization Point                 |
| SYNC                               | Converge threads after conditional branch |
| CAL                                | Relative Call                             |
| JCAL                               | Absolute Call                             |
| PRET                               | Pre-Return From Subroutine                |
| RET                                | Return From Subroutine                    |
| BRK                                | Break                                     |
| PBK                                | Pre-Break                                 |
| CONT                               | Continue                                  |
| PCNT                               | Pre-continue                              |
| EXIT                               | Exit Program                              |
| PEXIT                              | Pre-Exit                                  |
| BPT                                | BreakPoint/Trap                           |
| <b>Miscellaneous Instructions</b>  |   |
| NOP                                | No Operation                              |

| Opcode | Description                       |
|--------|-----------------------------------|
| CS2R   | Move Special Register to Register |
| S2R    | Move Special Register to Register |
| B2R    | Move Barrier To Register          |
| BAR    | Barrier Synchronization           |
| R2B    | Move Register to Barrier          |
| VOTE   | Vote Across SIMD Thread Group     |

# Chapter 5.

## NVPRUNE

**nvprune** prunes host object files and libraries to only contain device code for the specified targets.

### 5.1. Usage

**nvprune** accepts a single input file each time it's run, emitting a new output file. The basic usage is as following:

```
nvprune [options] -o <outfile> <infile>
```

The input file must be either a relocatable host object or static library (not a host executable), and the output file will be the same format.

Either the `--arch` or `--generate-code` option must be used to specify the target(s) to keep. All other device code is discarded from the file. The targets can be either a `sm_NN` arch (cubin) or `compute_NN` arch (ptx).

For example, the following will prune `libcublas_static.a` to only contain `sm_35` cubin rather than all the targets which normally exist:

```
nvprune -arch sm_35 libcublas_static.a -o libcublas_static35.a
```

Note that this means that `libcublas_static35.a` will not run on any other architecture, so should only be used when you are building for a single architecture.

### 5.2. Command-line Options

[Table 8](#) contains supported command-line options of **nvprune**, along with a description of what each option does. Each option has a long name and a short name, which can be used interchangeably.

Table 8 `nvprune` Command-line Options

| Option (long)   | Option (short)        | Description  |
|---|-----------------------|--|
| <code>--arch &lt;gpu architecture name&gt;,...</code> | <code>-arch</code>    | Specify the name of the NVIDIA GPU architecture which will remain in the object or library.  |
| <code>--generate-code</code>                          | <code>-gencode</code> | This option is same format as <code>nvcc --generate-code</code> option, and provides a way to specify multiple architectures which should remain in the object or library. Only the 'code' values are used as targets to match. Allowed keywords for this option: 'arch','code'. |
| <code>--output-file</code>                            | <code>-o</code>       | Specify name and location of the output file.  |
| <code>--help</code>                                   | <code>-h</code>       | Print this help information on this tool.  |
| <code>--options-file &lt;file&gt;,...</code>          | <code>-optf</code>    | Include command line options from specified file.  |
| <code>--version</code>                                | <code>-V</code>       | Print version information on this tool.  |

## **Notice**

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

## **Trademarks**

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

## **Copyright**

© 2015 NVIDIA Corporation. All rights reserved.