cuobjdump

DA-05536-001_v04 | September 2012

Application Note

DOCUMENT CHANGE HISTORY

DA-05536-001_v04

Version	Date	Authors	Description of Change
--	August 25, 2010	CUDA	Limited release.
01	October 15, 2010	CUDA	Added Instruction set for Fermi GPUs.
02	January 13, 2011	CUDA	Updated format for valid destination and source locations for GT200 and the Fermi instruction sets.
03	January 21, 2011	CUDA	Updated Overview section and added list of supported binary formats by platform.
04	September 13, 2012	CUDA	Updated supported options. Added Instruction set for Kepler GPUs.

TABLE OF CONTENTS

cuobjdump

OVERVIEW

NVIDIA® CUDA™ provides **cuobjdump,** a tool similar to the Linux command-line tool **objdump**. It extracts information from CUDA object files created by the NVIDIA® CUDA™ compiler **nvcc** and presents them in human readable format. CUDA binaries generated by the NVIDIA® CUDA™ compiler **nvcc** are referred to as **cubins**. The **cuobjdump** tool displays the assembly instructions for a particular kernel, making it useful for optimization and debugging by advanced users.

Table 1 contains a list of supported binary formats by platform.

Table 1. Supported Formats by Platform

	Linux	Windows	Mac
Executable Binary	Yes	Yes	No
Non-executable cubin Binary	Yes	Yes	Yes

This document contains a description of the various options supported by **cuobjdump**, and instruction sets supported on the GT200, Fermi and Kepler GPUs.

SUPPORTED OPTIONS

Table 2 contains the supported options to **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. Supported cuobjdump Options

Option(long)	Option(short)	Description
--all-fatbin	-all	Dump all fatbin sections. By default will only dump contents of executable fatbin (if exists), else relocatable fatbin if no executable fatbin.
--dump-cubin	-cubin	Dump cubin for all listed device functions.
--dump-elf	-elf	Dump ELF Object sections.
--dump-elf-symbols	-symbols	Dump ELF symbol names.
--dump-function-names	-fnam	Dump names of device functions. This option is implied if options --dump-sass, --dump-cubin or --dump-ptx are also given.
--dump-ptx	-ptx	Dump PTX for all listed device functions.
--dump-sass	-sass	Dump assembly for all listed device functions.
--file <filename>,...	-f	Specify names of source files whose fat binary structures must be dumped. Source files may be specified by the full path by which they were compiled using nvcc, or file name only (omitting the directory part), of file base name only (omitting directory and the '.cu' file extension).
--function <function name>,...	-fun	Specify names of device functions whose fat binary structures must be dumped.
--help	-h	Print this help information on this tool.
--options-file <file>,...	-optf	Include command line options from specified file.
--sort-functions	-sort	Sort functions when dumping sass.
--version	-V	Print version information on this tool.

GT200 INSTRUCTION SET

When the **-sass** option is used on cubins containing code for Compute 1.x devices, the following instructions are output by **cuobjdump** in the following 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] for constant memory
- local[X] for local memory

Table 3 lists valid instructions for the GT200 GPUs.

Table 3. 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
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

Opcode	Description
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
ISET	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

Opcode	Description
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

FERMI INSTRUCTION SET

When the **-sass** option is used on cubins containing code for Compute 2.x devices, the following instructions are output by **cuobjdump** in the following format:

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

Valid destination and source locations include:

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

Table 4 lists valid instructions for the Fermi GPUs.

Table 4. Fermi Instruction Set

Opcode	Description
Floating Point Instructions	
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
Integer Instructions	
IMAD	Integer Multiply Add

Opcode	Description
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
ISET	Integer Set
ISETP	Integer Set Predicate
ICMP	Integer Compare and Select
POPC	Population Count
Conversion Instructions	
Opcode	**Description**
F2F	Float to Float
F2I	Float to Integer
I2F	Integer to Float
I2I	Integer to Integer
Movement Instructions	
Opcode	**Description**
MOV	Move
SEL	Conditional Select/Move
PRMT	Permute
Predicate/CC Instructions	
Opcode	**Description**
P2R	Predicate to Register
R2P	Register to Predicate
CSET	CC Set

Opcode	Description
CSETP	CC Set Predicate
PSET	Predicate Set
PSETP	Predicate Set Predicate
Texture Instructions	
TEX	Texture Fetch
TLD	Texture Load
TLD4	Texture Load 4 Texels
TXQ	Texture Query
Compute Load/Store Instructions	
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
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
Surface Memory Instructions	
SULD	Surface Load
SULEA	Surface Load Effective Address

Opcode	Description
SUST	Surface Store
SURED	Surface Reduction
SUQ	Surface Query
Control Instructions	
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
Miscellaneous Instructions	
NOP	No Operation
S2R	Special Register to Register
B2R	Barrier to Register
LEPC	Load Effective PC
BAR	Barrier Synchronization
VOTE	Query condition across threads

KEPLER INSTRUCTION SET

When the **-sass** option is used on cubins containing code for Compute 3.x devices, the following instructions are output by **cuobjdump** in the following format:

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

Valid destination and source locations include:

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

Table 5 lists valid instructions for the Kepler GPUs.

Table 5. Kepler Instruction Set

Opcode	Description
Floating Point Instructions	
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
DMNMX	FP64 Minimum/Maximum
DSET	FP64 Set
DSETP	FP64 Set Predicate
Integer Instructions	

Opcode	Description
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
ISET	Integer Set
ISETP	Integer Set Predicate
ICMP	Integer Compare and Select
POPC	Population Count
Conversion Instructions	
Opcode	**Description**
F2F	Float to Float
F2I	Float to Integer
I2F	Integer to Float
I2I	Integer to Integer
Movement Instructions	
Opcode	**Description**
MOV	Move
SEL	Conditional Select/Move
PRMT	Permute
SHFL	Warp Shuffle
Predicate/CC Instructions	

Opcode	Description
Opcode	**Description**
P2R	Predicate to Register
R2P	Register to Predicate
CSET	CC Set
CSETP	CC Set Predicate
PSET	Predicate Set
PSETP	Predicate Set Predicate
Texture Instructions	
TEX	Texture Fetch
TLD	Texture Load
TLD4	Texture Load 4 Texels
TXQ	Texture Query
Compute Load/Store Instructions	
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
Surface Memory Instructions	
SUCLAMP	Surface Clamp
SUBFM	Surface Bit Field Merge
SUEAU	Surface Effective Address

Opcode	Description
SULDGA	Surface Load Generic Address
SUSTGA	Surface Store Generic Address
Control Instructions	
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
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
Miscellaneous Instructions	
NOP	No Operation
S2R	Special Register to Register
B2R	Barrier to Register
BAR	Barrier Synchronization
VOTE	Query condition across threads

Notice

OpenCL

Trademarks

Copyright