阿里云 中国站 控制台 备案 邮箱 登录
全部导航 最新活动 HOT 产品 解决方案 数据 · 智能 安全 云市场 支持 合作伙伴 免费注册
帮助与文档 请用关键词进行搜索，例如“服务器密码重置” 搜全部 搜本产品
搜索热词：远程连接服务器 挂载数据盘 域名解析 域名实名认证 账号实名认证 忘记密码
文档与视频 新手学堂 自助工具 联系客服 我的收藏
全部产品 高性能计算 产品简介 产品定价 快速入门 用户指南 最佳实践 开发指南 API参考 优化工具 kepler汇编器 概述
高性能计算 > 开发指南 > 优化工具 > kepler汇编器 > 使用手册
本文导读目录 点我提问

使用手册

更新时间：2017-06-07 13:26:11 分享：

Kepler Assembler is an almost full-featured assembler created for NVidia's Kepler Architecture. It can be used to generate binary files
assembly code, or to dump "sass" from a given cubin. Except for some kernels' descriptive information, including parameter number
the "sass" format supported by this tool is definitely the same as those dumped from "cuobjdump". The function of dumping "sass" c
cubin is useful, since it is difficult for most to write whole assembly code from scratch. Thus, dumping assembly code from a cubin co
tuning them, then generating cubin would be a more convenient, efficient and acceptable way to use this tool.

Just like asfermi and maxas, we wrote this tool to get extremely high device utility. The tools provided by NVidia didn't show enough s
kinds of needs in processes of pursuing ultimate performance. With Kepler assembler, it becomes possible to control kernel's register
adjust executing order of instructions, and even to change the scheduling details of a kernel.

From Fermi to Maxwell, NVidia has brought a number of changes to every new architecture generation, for example, new instruction s
new ISA encoding, and so on. Therefore, such kind of assembler could not equip with enough backward compatibility. Maxas cannot
Architecture, and Kepler assembler won't work at Fermi architecture, vice vasa.

2. Input Assembly File Format

If you want to generate a cubin file with your own self-turning assembly, you should learn to write your kernel with "sass" as below first.

Here is an example to illustrate what you can input to Kepler assembler to generate a cubin.

```
Kernel number: 1
<KernelName: yourKernelName
<Para__yourKernelName: num|4 size|32
<ParaDetail__yourKernelName: para1|8,para2|8,para3|8,para4|8
<Shared__yourKernelName: size|400 align|16
<Reg__yourKernelName: 13
<code:
/*0008*/    CTL: 00101100        MOV R1, c[0x0][0x44];
/*0010*/    CTL: 00101000        S2R R0, SR_CTAID.X;
/*0018*/    CTL: 00101000        MOV32I R12, 0x4;
/*0020*/    CTL: 00101000        S2R R3, SR_TID.X;
/*0028*/    CTL: 00100011        IMAD R2, R0, c[0x0][0x28], R3;
/*0030*/    CTL: 00000100        SHF.L.W R5, RZ, 0x2, R2;
/*0038*/    CTL: 00000000        IADD R8.CC, R5, c[0x0][0x148];
  ... ...
/*1c88*/    CTL: 00000000        STS [R5], R4;
/*1c90*/    CTL: 00000000        ST.E [R8], R0;
/*1c98*/    CTL: 00000000        EXIT;
/*1ca0*/    CTL: 00000000        BRA 0xe0;
/*1ca8*/    CTL: 00101110        NOP;
/*1cb0*/    CTL: 00000100        NOP;
/*1cb8*/    CTL: 00101110        NOP;
code>
```

You should begin your 'sass' file with a line to declare the number of kernels that you want to include in your cubin file.

`Kernel number: 10`

means your cubin file will have 10 CUDA kernels.

You can use `<` to start a line declaring particular kernel information.

`<KernelName: yourKernelName`

The line above points out current kernel name is `yourKernelName` .

And you should use lines that start with `<Para` and `<ParaDetail` to give the assembler current kernel's parameter information, including parameter number, total parameter size, and size of each parameter. For example, if your kernel in cuda C format is

`__global__ void yourKernelName(float *yourPara1, float yourPara2, int* yourPara3, bool yourPara4)`

Then you should write

```
<KernelName: yourKernelName
<Para__yourKernelName: num|4 size|32
<ParaDetail__yourKernelName: para1|8,para2|4,para3|8,para4|1
```

The number after `|` , indicates value of corresponding variable. For example, `num|4` means current kernel has 4 parameter, `size|32` total parameter size is 32 byte. In the line of `<ParaDetail__` , `para1|8` means the size of first parameter is 8 byte, `para2|4` means the second parameter is 4 byte, and so forth. Note that, in this line, the string before `|` , is fixed to be among `para1, para2 ... paraN` .

The lines with `<Shared__` illustrate kernel's shared memory usage. All shared memories that a kernel use should be taken into account, ir shared memory and dynamic shared memory (at compile time, in most cases, dynamic shared memory is zero). One thing should be mentic shared memory size should be set with the consideration of shared memory alignment, different types should obey different alignment rules

The lines with `<Reg__` will tell Kepler assembler how many registers the kernel will use per thread.

Every line between `<code:` and `code>` is made up of three parts. On left hand side of each line, `/*xxxx*/` indicates relative address instruction. The middle part is control code, which gives hints to schedulers on device to determine instruction-issue behaviors at run time. cuobjdump command to dump sass from a cubin, you can find some line of binary do not have corresponding assembly code.

```
code for sm_35
        Function : _Z5test1PfS_S_S_i
    .headerflags    @"EF_CUDA_SM35 EF_CUDA_PTX_SM(EF_CUDA_SM35)"
                                                    /* 0x08ac80a0a08c1000 */
    /*0008*/            MOV R1, c[0x0][0x44];                   /* 0x64c03c00089c0006 */
    /*0010*/            S2R R0, SR_CTAID.X;                     /* 0x86400000129c0002 */
    /*0018*/            MOV32I R4, 0x4;                         /* 0x74000000021fc012 */
    /*0020*/            S2R R3, SR_TID.X;                       /* 0x86400000109c000e */
    /*0028*/            IMAD R5, R0, c[0x0][0x28], R3;          /* 0x51080c00051c0016 */
    /*0030*/            IMAD R2.CC, R5, R4, c[0x0][0x150];      /* 0x910c10002a1c140a */
    /*0038*/            MOV R0, c[0x0][0x160];                  /* 0x64c03c002c1c0002 */
                                                    /* 0x088c10a010b010a0 */
    /*0048*/            IMAD.HI.X R3, R5, R4, c[0x0][0x154];    /* 0x931810002a9c140e */
    /*0050*/            IMAD R6.CC, R5, R4, c[0x0][0x158];      /* 0x910c10002b1c141a */
```

As shown above, `0x08ac80a0a08c1000` and `0x088c10a010b010a0` are control codes. Each of them can influence scheduling bel instructions following them. In Kepler assembler, we've divided every control line into 7 parts, put them ahead of every single instruction line assembler, `CTL xxxxxxxx` part is what we are talking about. It could determine how two instructions (the one right after it, and the one in i be issued. At this point, we have done some experiments, and tried to understand how control code works, but there are still a lot of unsolv We are glad to discuss and communicate with anyone who is interested in it.

The third part of a code line is instruction, which will be issued and executed on processor at run time. The details about instruction format displayed in section 5.

3. Use Kepler Assembler

NVidia provides the CUDA module API. This API enables us to load CUDA binary and launch pre-compiled kernels at host, give us a practic our own hand-tuning kernels.

Here, we give a completed example to show how you can use this assembler to tune your kernel. This example will start from a CUDA kern CUDA C language. Then, this kernel will be complied by nvcc to generate a cubin. Kepler assembler helps to convert this cubin file into a as file, in which you can do some changes. When all optimization processes complete, assembly code file could be input to Kepler assembler a cubin. After all of these above, this cubin will be used in a sample C program through CUDA module.

This is a quite simple kernel: two matrices of n by n are input, every element of each matrix is multiplied by itself 12 times, results are summ whole column and restored to an array. Well, this kernel appears a little kooky. The reason we use it is to give an example whose performan by computing. We suggest this assembler won't be used until you have done all of regular optimizations, since writing and editing assembly typically not as easy as programing with CUDA C, and performance issues involved with factors such as memory bandwidth could be solve redesigning algorithms.

```
__global__ void testFunction(float* matrix1, float* matrix2, float* output_array1, float* output_array2, int n){
  int idx = blockIdx.x*blockDim.x+threadIdx.x;
  float dataFromM1, dataFromM2;
  output_array1[idx] = 0.0;
  output_array2[idx] = 0.0;
  for(int i = 0; i < n; i++){
    dataFromM1 = matrix1[idx+i*n];
    dataFromM2 = matrix2[idx+i*n];
    output_array1[idx] += dataFromM1*dataFromM1*dataFromM1*dataFromM1*dataFromM1*dataFromM1*dataFromM1*dataFromM1*dataFromM1*dataFromM1*da
    output_array2[idx] += dataFromM2*dataFromM2*dataFromM2*dataFromM2*dataFromM2*dataFromM2*dataFromM2*dataFromM2*dataFromM2*da
```

本文导读目录

点我提问

```
  }
}
```

Save this function in test.cu, and compile it to generate cubin file with sm_35 architecture.

```
nvcc -gencode arch=compute_35,code=sm_35 -cubin test.cu
```

Now we have test.cubin, which is an elf file. Upload this cubin to our website, the assembly code corresponding to it will be dumped as foll use `#` to begin an annotation line.

```
Kernel number: 1
#Current Kernel Name. Note: kernel name should be make up from letters, digits and the underscore character.
<KernelName: _Z12testFunctionPfS_S_S_i
#Parameter Number and Parameter Size.
<Para___Z12testFunctionPfS_S_S_i: num|5 size|36
#Size of every parameter particularly. Note: 'para1' represents the first parameter, 'para2' represents the second, and so forth.
<ParaDetail___Z12testFunctionPfS_S_S_i: para1|8,para2|8,para3|8,para4|8,para5|4
#Register Number per Thread.
<Reg___Z12testFunctionPfS_S_S_i: 16
<code:
#'CTL xxxxxxxx' part is control code, which determines instruction scheduling behavior.

/*0008*/      CTL: 00000000       MOV R1, c[0x0][0x44];
/*0010*/      CTL: 00000100       S2R R0, SR_CTAID.X;
/*0018*/      CTL: 00100011       MOV32I R5, 0x4;
/*0020*/      CTL: 00101000       S2R R3, SR_TID.X;
/*0028*/      CTL: 00101000       IMAD R6, R0, c[0x0][0x28], R3;
/*0030*/      CTL: 00100000       IMAD R2.CC, R6, R5, c[0x0][0x150];
/*0038*/      CTL: 00101011       MOV R0, c[0x0][0x160];

/*0048*/      CTL: 00101000       IMAD.HI.X R3, R6, R5, c[0x0][0x154];
/*0050*/      CTL: 00000100       IMAD R4.CC, R6, R5, c[0x0][0x158];
/*0058*/      CTL: 00101100       ST.E [R2], RZ;
/*0060*/      CTL: 00000100       ISETP.LT.AND P0, PT, R0, 0x1, PT;
/*0068*/      CTL: 00101000       IMAD.HI.X R5, R6, R5, c[0x0][0x15c];
/*0070*/      CTL: 00000100       MOV R0, RZ;
/*0078*/      CTL: 00100011       ST.E [R4], RZ;

/*0088*/      CTL: 00101110       @P0 EXIT;
/*0090*/      CTL: 00101000       MOV R7, c[0x0][0x160];
/*0098*/      CTL: 00101100       ISETP.GT.AND P0, PT, R7, 0x1, PT;
/*00a0*/      CTL: 00000000       @!P0 BRA 0x390;
/*00a8*/      CTL: 00000100       MOV R7, c[0x0][0x160];
/*00b0*/      CTL: 00101000       MOV32I R8, 0x4;
/*00b8*/      CTL: 00000000       IADD R7, R7, -0x1;

/*00c8*/      CTL: 00101000       IMAD R9, R0, c[0x0][0x160], R6;
/*00d0*/      CTL: 00101100       IMAD R12.CC, R9, R8, c[0x0][0x140];
/*00d8*/      CTL: 00101000       IMAD.HI.X R13, R9, R8, c[0x0][0x144];
/*00e0*/      CTL: 00000100       LD.E R10, [R12];
/*00e8*/      CTL: 00101100       IMAD R14.CC, R9, R8, c[0x0][0x148];
/*00f0*/      CTL: 00101010       IMAD.HI.X R15, R9, R8, c[0x0][0x14c];
/*00f8*/      CTL: 00101000       FMUL R11, R10, R10;

/*0108*/      CTL: 00000100       LD.E R9, [R14];
/*0110*/      CTL: 00101000       FMUL R11, R11, R10;
/*0118*/      CTL: 00101000       FMUL R11, R11, R10;
/*0120*/      CTL: 00101000       FMUL R11, R11, R10;
/*0128*/      CTL: 00100000       FMUL R11, R11, R10;
/*0130*/      CTL: 00100111       FMUL R12, R9, R9;
/*0138*/      CTL: 00100000       FMUL R11, R11, R10;

/*0148*/      CTL: 00100111       FMUL R12, R12, R9;
/*0150*/      CTL: 00100000       FMUL R11, R11, R10;
/*0158*/      CTL: 00100111       FMUL R12, R12, R9;
/*0160*/      CTL: 00100000       FMUL R11, R11, R10;
/*0168*/      CTL: 00100111       FMUL R13, R12, R9;
/*0170*/      CTL: 00100000       FMUL R11, R11, R10;
/*0178*/      CTL: 00100000       LD.E R12, [R2];

/*0188*/      CTL: 00100110       FMUL R13, R13, R9;
/*0190*/      CTL: 00100001       FMUL R11, R11, R10;
/*0198*/      CTL: 00100110       FMUL R13, R13, R9;
/*01a0*/      CTL: 00100001       FMUL R11, R11, R10;
/*01a8*/      CTL: 00100110       FMUL R13, R13, R9;
/*01b0*/      CTL: 00101000       FMUL R11, R11, R10;
/*01b8*/      CTL: 00100000       FFMA R10, R11, R10, R12;

/*01c8*/      CTL: 00000100       FMUL R12, R13, R9;
/*01d0*/      CTL: 00101000       IADD R11, R0, 0x1;
/*01d8*/      CTL: 00000100       FMUL R12, R12, R9;
/*01e0*/      CTL: 00101000       ST.E [R2], R10;
```

本文导读目录

点我提问

```
/*01e8*/      CTL: 00100101        FMUL R12, R12, R9;
/*01f0*/      CTL: 00100010        LD.E R13, [R4];
/*01f8*/      CTL: 00100000        FMUL R12, R12, R9;

/*0208*/      CTL: 00000100        IMAD R11, R11, c[0x0][0x160], R6;
/*0210*/      CTL: 00101000        IADD R0, R0, 0x2;
/*0218*/      CTL: 00000100        FMUL R12, R12, R9;
/*0220*/      CTL: 00101010        ISETP.LT.AND P0, PT, R0, R7, PT;
/*0228*/      CTL: 00100000        FFMA R9, R12, R9, R13;
/*0230*/      CTL: 00100111        IMAD R12.CC, R11, R8, c[0x0][0x140];
/*0238*/      CTL: 00100100        ST.E [R4], R9;

/*0248*/      CTL: 00101001        IMAD.HI.X R13, R11, R8, c[0x0][0x144];
/*0250*/      CTL: 00000100        LD.E R10, [R12];
/*0258*/      CTL: 00101100        IMAD R14.CC, R11, R8, c[0x0][0x148];
/*0260*/      CTL: 00101010        IMAD.HI.X R15, R11, R8, c[0x0][0x14c];
/*0268*/      CTL: 00101000        FMUL R9, R10, R10;
/*0270*/      CTL: 00101000        FMUL R9, R9, R10;
/*0278*/      CTL: 00101000        FMUL R11, R9, R10;

/*0288*/      CTL: 00000100        LD.E R9, [R14];
/*0290*/      CTL: 00101000        FMUL R11, R11, R10;
/*0298*/      CTL: 00101110        FMUL R11, R11, R10;
/*02a0*/      CTL: 00100000        FMUL R12, R9, R9;
/*02a8*/      CTL: 00100111        FMUL R11, R11, R10;
/*02b0*/      CTL: 00100000        FMUL R12, R12, R9;
/*02b8*/      CTL: 00100111        FMUL R11, R11, R10;

/*02c8*/      CTL: 00100000        FMUL R12, R12, R9;
/*02d0*/      CTL: 00100111        FMUL R11, R11, R10;
/*02d8*/      CTL: 00100000        FMUL R13, R12, R9;
/*02e0*/      CTL: 00100000        FMUL R11, R11, R10;
/*02e8*/      CTL: 00100110        LD.E R12, [R2];
/*02f0*/      CTL: 00100000        FMUL R13, R13, R9;
/*02f8*/      CTL: 00100111        FMUL R11, R11, R10;

/*0308*/      CTL: 00100000        FMUL R13, R13, R9;
/*0310*/      CTL: 00100111        FMUL R11, R11, R10;
/*0318*/      CTL: 00100000        FMUL R13, R13, R9;
/*0320*/      CTL: 00101000        FMUL R11, R11, R10;
/*0328*/      CTL: 00101000        FFMA R10, R11, R10, R12;
/*0330*/      CTL: 00000100        FMUL R12, R13, R9;
/*0338*/      CTL: 00101000        ST.E [R2], R10;

/*0348*/      CTL: 00100101        FMUL R12, R12, R9;
/*0350*/      CTL: 00100010        LD.E R13, [R4];
/*0358*/      CTL: 00101000        FMUL R12, R12, R9;
/*0360*/      CTL: 00101000        FMUL R12, R12, R9;
/*0368*/      CTL: 00101000        FMUL R12, R12, R9;
/*0370*/      CTL: 00101000        FFMA R9, R12, R9, R13;
/*0378*/      CTL: 00100000        ST.E [R4], R9;

/*0388*/      CTL: 00101110        @P0 BRA 0xc0;
/*0390*/      CTL: 00101100        ISETP.LT.AND P0, PT, R0, c[0x0][0x160], PT;
/*0398*/      CTL: 00101110        @!P0 EXIT;
/*03a0*/      CTL: 00000100        MOV32I R7, 0x4;
/*03a8*/      CTL: 00100000        NOP;
/*03b0*/      CTL: 00100000        NOP;
/*03b8*/      CTL: 00000000        NOP;

/*03c8*/      CTL: 00000100        IMAD R8, R0, c[0x0][0x160], R6;
/*03d0*/      CTL: 00101000        IADD R0, R0, 0x1;
/*03d8*/      CTL: 00101100        IMAD R12.CC, R8, R7, c[0x0][0x140];
/*03e0*/      CTL: 00101000        IMAD.HI.X R13, R8, R7, c[0x0][0x144];
/*03e8*/      CTL: 00000100        LD.E R9, [R12];
/*03f0*/      CTL: 00101110        ISETP.LT.AND P0, PT, R0, c[0x0][0x160], PT;
/*03f8*/      CTL: 00101000        IMAD R12.CC, R8, R7, c[0x0][0x148];

/*0408*/      CTL: 00101000        FMUL R10, R9, R9;
/*0410*/      CTL: 00100000        FMUL R10, R10, R9;
/*0418*/      CTL: 00101000        IMAD.HI.X R13, R8, R7, c[0x0][0x14c];
/*0420*/      CTL: 00000100        FMUL R10, R10, R9;
/*0428*/      CTL: 00101000        LD.E R8, [R12];
/*0430*/      CTL: 00101000        FMUL R10, R10, R9;
/*0438*/      CTL: 00100101        FMUL R10, R10, R9;

/*0448*/      CTL: 00100010        FMUL R11, R8, R8;
/*0450*/      CTL: 00100101        FMUL R10, R10, R9;
/*0458*/      CTL: 00100010        FMUL R11, R11, R8;
/*0460*/      CTL: 00100101        FMUL R10, R10, R9;
/*0468*/      CTL: 00100010        FMUL R11, R11, R8;
/*0470*/      CTL: 00100101        FMUL R10, R10, R9;
```

本文导读目录

点我提问

```
/*0478*/      CTL: 00100010        FMUL R11, R11, R8;

/*0488*/      CTL: 00100101        FMUL R10, R10, R9;
/*0490*/      CTL: 00100010        FMUL R11, R11, R8;
/*0498*/      CTL: 00100000        FMUL R12, R10, R9;
/*04a0*/      CTL: 00100100        LD.E R10, [R2];
/*04a8*/      CTL: 00100010        FMUL R11, R11, R8;
/*04b0*/      CTL: 00100101        FMUL R12, R12, R9;
/*04b8*/      CTL: 00100010        FMUL R11, R11, R8;

/*04c8*/      CTL: 00101000        FMUL R12, R12, R9;
/*04d0*/      CTL: 00101000        FFMA R10, R12, R9, R10;
/*04d8*/      CTL: 00000100        FMUL R9, R11, R8;
/*04e0*/      CTL: 00101000        ST.E [R2], R10;
/*04e8*/      CTL: 00101000        FMUL R9, R9, R8;
/*04f0*/      CTL: 00000100        FMUL R11, R9, R8;
/*04f8*/      CTL: 00101000        LD.E R9, [R4];

/*0508*/      CTL: 00101000        FMUL R11, R11, R8;
/*0510*/      CTL: 00101000        FMUL R12, R11, R8;
/*0518*/      CTL: 00101000        FFMA R8, R12, R8, R9;
/*0520*/      CTL: 00000100        ST.E [R4], R8;
/*0528*/      CTL: 00101110        @P0 BRA 0x3c0;
/*0530*/      CTL: 00000100        MOV RZ, RZ;
/*0538*/      CTL: 00101110        EXIT;
/*0540*/      CTL: --------      BRA 0x540;
/*0548*/      CTL: --------      NOP;
/*0550*/      CTL: --------      NOP;
/*0558*/      CTL: --------      NOP;
/*0560*/      CTL: --------      NOP;
/*0568*/      CTL: --------      NOP;
/*0570*/      CTL: --------      NOP;
/*0578*/      CTL: --------      NOP;

code>
```

Now you can play with this kernel through assembly code. Try to change some control codes, or move one instruction to another line
change some control code in a few lines.

Replace

```
/*0148*/      CTL: 00100111        FMUL R12, R12, R9;
/*0150*/      CTL: 00100000        FMUL R11, R11, R10;
/*0158*/      CTL: 00100111        FMUL R12, R12, R9;
/*0160*/      CTL: 00100000        FMUL R11, R11, R10;
/*0168*/      CTL: 00100111        FMUL R13, R12, R9;
/*0170*/      CTL: 00100000        FMUL R11, R11, R10;
/*0178*/      CTL: 00100000        LD.E R12, [R2];
```

with

```
/*0148*/    CTL: 00000101        FMUL R12, R12, R9;
/*0150*/    CTL: 00000100        FMUL R11, R11, R10;
/*0158*/    CTL: 00000101        FMUL R12, R12, R9;
/*0160*/    CTL: 00000100        FMUL R11, R11, R10;
/*0168*/    CTL: 00000101        FMUL R13, R12, R9;
/*0170*/    CTL: 00000100        FMUL R11, R11, R10;
/*0178*/    CTL: 00100000        LD.E R12, [R2];
```

When your assembly file is ready, you can use Kepler assembler to generate the corresponding cubin, download this cubin and congratulat
have a self-optimized cubin file. In next section, we will show you how to use this cubin in your own project.

4. Use CUBIN in Your Program

We use host-end CUDA module API to load cubin and launch kernels in a cubin. Here is an example of using the cubin file generated in pre
Before you could launch a particular kernel in your program, you should load CUDA module and get this kernel from cubin.

```
 CUdevice cuDevice;
 CUcontext cuContext;
 CUmodule cuModule;

 CUresult error;
 cuInit(0);
 int devID = 0;

 // get device
 error = cuDeviceGet(&cuDevice, devID);
 if (error != CUDA_SUCCESS){
   std::cout<<" cuDeviceGet error! Error code: "<<error<<std::endl;
 }

 // create context
 error = cuCtxCreate(&cuContext, 0, cuDevice);
 if (error != CUDA_SUCCESS){
   std::cout<<" cuCtxCreate error! Error: code: "<<error<<std::endl;
 }

 // load module
 error = cuModuleLoad(&cuModule, "test.cubin");
 if (error != CUDA_SUCCESS){
   std::cout<<" cuModuleLoad error! Error code: "<<error<<std::endl;
 }

 // get function
 CUfunction test1;
 error = cuModuleGetFunction(&test1, cuModule, "kernelNameinCubin");
 if (error != CUDA_SUCCESS){
   std::cout<<" cuModuleGetFunction error! Error code: "<<error<<std::endl;
 }
```

One thing, that we should mention here, is the kernel name inputted to cuModuleGetFunction should be the name of kernel in cubin file or a
the one in CUDA C program. Nvcc will change your kernel name when it compiles a program.

After input parameters are declared and initialized, you should use an array of pointers to wrap up their addresses.

`void *args[] = {&matrix1, &matrix2, &outputArray1, &outputArray2, &n};`

Then you can launch your kernel by lines below.

```
error = cuLaunchKernel(test1,  blockNum.x, blockNum.y, blockNum.z,
                       threadNum.x, threadNum.y, threadNum.z,
                       shareMemSize,
                       NULL, args, NULL);
if (error != CUDA_SUCCESS){
  std::cout<<"cuLaunchKernel error! Error code: "<<error<<std::endl;
}
```

本文导读目录

点我提问

In this way, you can create your project and run your hand writing cuda kernels on Kepler architecture. Frankly, tuning kernel through
typically a challenging job and needs some experiments to unveil all kinds of details about Kepler device. But if you want to make you
efficient, and you've tried every method to optimize your code. You should take this assembler and give it a shoot. After you reduce s
or relocate some instructions in your kernel to make them issued with less stall. You may get significant performance improvement.

The only purpose of this example above is to walk you through the whole process of using kepler assembler. We did not try to do any deep
Even so, after simply changing those control codes mentioned above, we could still accelerate this kernel by about 0.5us.

5. Kepler Instruction Set

operand

In kepler native assembler, operand mainly has three types:

1. register: register number ranges from 0 to 255(use RZ instead of R255);
2. immediate number: for float number, using decimal form; for integer number, using hexadecimal form;
3. constant memoory: constand memory has the form of c[imm1][imm2], where imm1 and imm2 are both hexadecimal number, imm1 refe
   number of constant memory, and imm2 refers to the offset to that bank;

From now, we will use composite_operand in the filed where all three types are all OK. For example, the usage of FADD is:

`FADD rd, r1, composite_opreand;`

which means all three forms below are OK:

```
FADD rd, r1, r2;
FADD rd, r1, c[][];
FADD rd, r1, imm;
```

there are also other kinds of register, for example:pridectied register(ranged from 0 to 7(using PT instead of P7))

1.FFMA

Description:

FP32 Fused Multiply Add, has two instructions:FFMA32I and FFMA:
FFMA32I: the immediate number is ieee-754 compatibled, which means it has 23-bit mantissa in the ieee-754 form; FFMA: the immediate n

ieee-754 partial compatibled, it only has 11-bit mantissa in the ieee-754 form;

Usage:

```
FFMA32I(.SAT)(.FTZ)(.FMZ) rd(.CC), (-)r1, imm, (-)r3;
FFMA(.rnd)(.FMZ)(.FTZ)(.SAT) rd(.CC), (-)r1, composite_operand, (-)r3;
FFMA(.rnd)(.FMZ)(.FTZ)(.SAT) rd(.CC), (-)r1, r2, (-)c[][];
```

Others:

In FFMA32I instruction, r3 must be the same with rd
.rnd refers to the round mode, it can be one of RD, RZ, RP, and RM; If not specify, default round mode is RN;
.CC refers to set the carry bit

2.FADD

Description:

FP32 Add, also has two instructions:FADD32I and FADD

Usage:

```
FADD32I(.FTZ)  rd(.CC), (-)(|)r1(|), imm(.NEG)
FADD(.rnd)(.SAT)(.FTZ) rd(.CC), (-)(|)r1(|), (-)(|)composite_operand(.ABS)(.NEG)(|);
```

本文导读目录

点我提问

Others:

when composite_operand refer to immediate number, use (.ABS) and (.NEG) flags instead of (-) and (||)

3.FCMP

Description:

FP32 compare

Usage:

```
FCMP(.compare_op)(.FTZ) rd, r1, composite_operand, r3;
FCMP(.compare_op)(.FTZ) rd, r1, r2, c[][];
```

Others

compare_op refer to one of GT, NE, GE, NUM, NAN, LTU, EQU, and LEU

4.FMUL

Description:

FP32 multiply, has two instructions: FMUL32I and FMUL

Usage:

```
FMUL32I(.FMZ)(.FTZ)(.SAT) rd(.CC), imm;
FMUL(.rnd)(.SAT)(.FMZ)(.FTZ)(.M8)(.D4)(.D2) rd(.CC), (-)r1, composite_operand;
```

Others:

5.FMNMX

Description:

FP32 minimum/maximum

Usage:

`FMNMX(.FTZ) rd(.CC), (-)(|)r1(|), (-)(|)composite_operand(|)(.ABS)(.NEG), (!)PT;`

Others:

PT refers to MAX while !PT refers to MIN

6.FSWZ

Description:

FP32 swizzle

Usage:

```
FSWZ(.rnd)(.FTZ)(.mode)(.NDV) rd(.CC), r1, r2, PPPPPPPP;
```

Others:

.mode refers to one of .0000, .1111, .2222, .3333, .1032, .2301

7.FSET

Description:

FP32 set

Usage:

```
FSET(.FTZ)(.BF).compare_op.logic_op rd(.CC), (-)(|)r1(|), (-)(|)composite_operand(|)(.ABS)(.NEG);
```

Others:

logic_op refer to one of AND, OR, and XOR
compare_op refer to one of [F, LT, EQ, LE, GT, NE, GE, NUM, NAN, LTU, EQU, LEU, GTU,NEU, GEU, T]

8.FSETP

Description:

FP32 set predicate

Usage:

```
FSETP(.FTZ)(.compare_op)(.logic_op) p1, p2, (-)(|)r1(|), (-)(|)composite_operand(|)(.ABS)(.NEG);
```

Others:

compare_op and logic_op are the same with FSET

9.FCHK

Description:

FP32 division test

Usage:

```
FCHK.divide p1, (-)(|)r1(|), (-)(|)composite_operand(|)(.ABS)(.NEG);
```

Others:

10.RRO

Description:

FP range reduction operator

Usage:

```
RRO.SINCOS(.EX2) rd, (-)(|)composite_operand(|)(.ABS)(.NEG);
```

Others:

11.MUFU

Description:

FP multi-function opreator

Usage:

```
MUFU.function_type(.SAT) rd, (-)(|)r1(|);
```

Others:

function_type refers to one of [COS, SIN, EX2, LG2, RCP, RSQ, RCP64, RSQ64]

12.DFMA

Description:

FP64 fused multiply add

Usage:

```
DFMA(.rnd) rd(.CC), (-)r1, composite_operand, (-)r3;
DFMA(.rnd) rd(.CC), (-)rd, r2, (-)c[][];
```

Others:

13.DADD

Description:

FP64 add

Usage:

```
DADD(.rnd) rd(.CC), (-)(|)r1(|), (-)(|)composite_operand(|)(.ABS)(.NEG);
```

Others:

14.DMUL

Description:

FP64 multiply

Usage:

```
DMUL(.rnd) rd(.CC) (-)r1, composite_operand;
```

Others:

15.DMNMX

Description:

FP64 minimum/maximum

Usage:

```
DMNMX rd(.CC), (-)(|)r1(|), (-)(|)composite_operand(|)(.ABS)(.NEG), p1;
```

Others:

16.DSET

Description:

FP64 set

Usage:

```
DSET.compare_op.logic_op(.BF) rd(.CC), (-)(|)r1(|), (-)(|)composite_oprand(|)(.ABS)(.NEG), (!)p1;
```

Others:

compare_op and logic_op are the same with FSET

17.DSETP

Description:

FP64 set predicate

Usage:

```
DSETP.compare_op.logic_op p1, p2, (-)(|)r1(|), (-)(|)composite_operand(|)(.ABS)(NEG), (!)p3;
```

Others:

18.IMAD

Description:

Integer multiply added; has two instructions:IMAD32I and IMAD

Usage:

```
IMAD32I(.HI)(.d_type)(.s_type)(.PO) rd(.CC), (-)r1, imm, (-)r3;
IMAD(.HI)(.d_type)(.s_type)(.PO)(.X)(.SAT) rd(.CC), (-)r1, composite_operand, (-)r3;
IMAD(.HI)(.d_type)(.s_type)(.PO)(.X)(.SAT) rd(.CC), (-)r1, r2, (-)c[][];
```

本文导读目录
点我提问

Others:

d_type is one of [U32 and S32];s_type is one of [U32 and S32];
if both are not specified, both are default S32;if one is U32, then both must specify, and the first one is d_type and the second is s_type;
X means add the carry bit in the carry register

19.IMADSP

Description:

Integer extract multiply add

Usage:

```
IMADSP(.d_type)(.s1_type)(s2_type)(.SD) rd(.CC), r1, composite_operand, r3;
IMADSP(.d_type)(.s1_type)(s2_type)(.SD) rd(.CC), r1, r2, c[][];
```

Others:

d_type is one of [U32, U24, U16H0, U16H1, S32, S24, S16H0, S16H1];
s1_type is one of [U24, S24, U16H0, S16H0]
s2_type is one of [U32,U24,U16H0, S32, S24, S16H0]

20.IMUL

Description:

Integer multiply; has two instructions:IMUL32I and IMUL

Usage:

```
IMUL32I(.HI)(.d_type)(.s_type) rd(.CC), r1, imm;
IMUL(.Hi)(.d_type)(.s_type) rd(.CC), r1, composite_operand;
```

Others:

d_type is one of [S32, U32]
s_type is one of [S32, U32]
when both d_type and s_type are all S32, can hide both S32 flag

21.IADD

Description:

Integer add, has two instructions:IADD32I and IADD

Usage:

```
IADD32I(.SAT)(.X)(.P0) rd(.CC), (-)r1, imm;
IADD(.SAT)(.X)(.P0) rd(.CC), (-r1), (-)composite_operand;
```

Others:

22.ISCADD

Description:

Integer scaled add, has two instructions: ISCADD32I and ISCAD

Usage:

```
ISCADD32I rd(.CC), r1, imm1, imm2;
ISCAD(.P0) rd(.CC), (-)r1, (-)composite_opreand;
```

Others:

23.ISAD

Description:

Integer sum of abs diff

Usage:

```
ISAD(.type) rd(.CC), r1, composite_operand, r3;
ISAD(.type) rd(.CC), r1, r2, composite_operand;
```

Others:

type is one of [S32, U32]

24.IMNMX

Description:

Integer minimum/maximum

Usage:

`IMNMX(.type)(.mode) rd, r1, composite_operand, (!)PT;`

Others:

type is one of [S32, U32]
mode is one of [XLO, XMED, XHI]

25.BFE

Description:

Integer bit field extract

Usage:

`BFE(.BREW)(.type) rd(.CC), r1, composite_operand;`

Others:

type is one of [S32, U32]

26.BFI

Description:

Integer bit field insert

Usage:

```
BFI rd(.CC), r1, composite_operand, r3;
BFI rd(.CC), r1, r2, c[][];
```

Others:

27.SHR

Description:

Integer shift right

Usage:

`SHR(.type)(.W)(.mode) rd(.CC), r1, composite_operand;`

Others:

type is one of [S32, U32]
mode is one of [X, XHI]

28.SHL

Description:

Integer shift left

Usage:

`SHL(.type)(.W)(.mode) rd(.CC), r1, composite_operand;`

Others:

type is one of [S32, U32]
mode is one of [X, XHI]

29.SHF

Description:

Integer funnel shift

Usage:

```
SHF(.L)(.R)(.W)(.X)(.HI)(.S64)(.U64) rd(.CC), r1, imm, r3;
SHF(.L)(.R)(.W)(.X)(.HI)(.S64)(.U64) rd(.CC), r1, r2, r3;
```

Others:

30.LOP

Description:

Integer logic operation, has two instructions: LOP32I and LOP;

Usage:

```
LOP32I(.logic_op) rd(.CC), (-)r1, (-)imm;
LOP(.logic_op)(.X) rd(.CC), (-)r1, (-)composite+operand;
```

Others:

logic_op is one of [AND, OR, XOR, and PASS_B]

本文导读目录

点我提问

31.FLO

Description:

Integer find leading one

Usage:

`FLO(.type)(.SH) rd(.CC), (~)composite_operand(.INV);`

Others:

type is one of [U32, S32]

32.ISET

Description:

Integer set

Usage:

`ISET(.type)(.compare_op)(.logic_op)(.X)(.BF) rd(.CC), r1, composite_operand, (!)PT;`

Others:

type is one of [U32, S32]
logic_op is one of [AND, OR, and XOR]
compare_op is one of [F, LT, EQ, LE, GT, NE, GE, T]

33.ISETP

Description:

Integer set predicate

Usage:

`ISETP(.type)(compare_op)(logic_op)(.X) p0, p1, r1, composite_operand, (!)PT;`

Others:

the same with ISET

33.ICMP

Description:

Integer copare and select

Usage:

```
ICMP(.compare_op)(.type) rd, r1,composite_operand, r3;
ICMP(.compare_op)(.type) rd, r1, r2, c[][];
```

Others:

compare_op is the same with ISET

34.POPC

Description:

Population count

Usage:

```
POPC rd, (~)r1, (~)composite_operand(.INV);
```

Others:

35.F2F

Description:

Float to float

Usage:

```
F2F(.SAT)(.FTZ)(.rnd)(d_type)(s_type) rd(.CC), (-)(|)composite_operand(|)(.ABS)(.NEG)(.H1);
```

Others:

d_type is one of [F16, F32, and F64]
s_type is one of [F16, F32, and F64]

36.F2I

Description:

Float to integer

Usage:

```
F2I(.FTZ)(.mode)(d_type)(s_type) rd(.CC), (-)(|)composite_operand(|)(.ABS)(.NEG)(.H1)
```

Others:

mode is one of [FLOOR, CEIL, and TRUNC]
s_type is one of [F16, F32, and F64]
d_type is one of [U8, U16, U32, U64, S8, S16, S32, and S64]

37.I2F

Description:

Integer to float

Usage:

```
I2F(.rnd)(.d_type)(.s_type) rd(.CC), (-)(|)composite_operand(|)(.ABS)(.NEG)(.B_moed);
```

Others:

rnd is one of [RN, RM, RP, and RZ]
d_type is one of [F16, F32, and F64]
s_type is one of [U8, U16, U32, U64, S8, S16, S32, and S64]
B_mode is one of [B1, B2, and B3]

38.I2I

Description:

Integer to integer

Usage:

```
I2I(.rnd)(d_type)(.s_type) rd(.CC), (-)(|)composite_operand(|)(.ABS)(.NEG)(.B_mode);
```

Others:

rnd is one of [RN, RM, RP, and RZ]
d_type and s_type both are one of [U8, U16, U32, U64, S8, S16, S32, and S64]
B_mode is one of [B1, B2, and B3]

39.MOV

Description:

move, has two instructions: MOV32I and MOV

Usage:

```
MOV32I rd, imm;
MOV rd, composite_operand;
```

Others:

40.SEL

Description:

Conditional select/mov

Usage:

`SEL rd, r1, composite_operand, p1;`

Others:

41.PRMT

Description:

Permute

Usage:

`PRMT rd, r1, composite_operand, r3;`

Others:

42.SHFL

Description:

Warp shuffle

Usage:

```
SHFL(.mode) p1, rd, r1, imm,r3;
SHFL(.mode) p1, rd, r1, imm1, imm2;
```

Others:

mode is one of [IDX, UP, DOWN, and BFLY]

43.P2R

Description:

Predicate to register

Usage:

`P2R(.H1) rd, PR, r1, composite_operand;`

Others:

44.R2P

Description:

Register to predicate

Usage:

`R2P(.H1) PR, r1, composite_operand;`

Others:

45.CSET

Description:

CC set

Usage:

```
CSET(.compare_op)(.logic_op)(.BF) rd(.CC), CC, (!)p1;
```

Others:

logic_op is one of [AND, OR, and XOR]
compare_op is one of [F, LE, EQ, LE, GT, NE, GE, NE, GE, NUM, NAN, LTU, EQU, LEU, GTU, NEU, GEU, OFF, LO, SFF, LS, HI, SFT, HS, OFI
CSM_TR, CSM_MX, TCSM_TA, FCSM_TR, FCSM_TR, RLT, and RGT]

46.CSETP

Description:

CC set predicate

Usage:

```
CSETP(.compare_op)(.logic_op) p1, p2, CC, (!)p3;
```

Others: compare_op and logic_op are both the same as CSET

47.PSET

Description:

Predicate set

Usage:

```
PSET(.logic_op)(.logic_op)(.BF) rd(.CC), (!)p1, (!)p2, (!)p3;
```

Others:

logic_op is one of [AND, OR, and XOR]

48.PSETP

Description:

Predicate set predicate

Usage:

```
PSETP(.logic_op)(.logic_op) p1, p2, (!)p3, (!)p4, (!)p4;
```

Others:

logic_op is one of [AND, OR, and XOR]

49.TEX

Description:

Texture fetch

Usage:

```
TEX(.AFFI)(.DC)(.NDV)(.T)(.P)(.NODEP)(.mode) rd, r1, (r2), imm1, dim, imm2;
TEX.B(.AFFI)(.DC)(.NDV)(.T)(.P)(.NODEP)(.mode) rd, r1, (r2), 0x0, dim, imm2;
```

Others:

mode is one of [LZ, LB, LL]
dim is one of [1D, 2D, 3D, ARRARY_1D, ARRAY_2D, ARRAY_3D]
imm2 is a 4-bit size

50.TLD

Description:

Texture load

Usage:

```
TLD(.LL)(.MS)(.CL)(.AOFFI)(.T)(.P)(.NODEP)(.LZ) rd, r1, r2, imm1, dim, imm2
TLD.B(.LL)(.MS)(.CL)(.AOFFI)(.T)(.P)(.NODEP)(.LZ) rd, r1, r2, 0x0, dim, imm2
```

Others:

dim is one of [1D, 2D, 3D, ARRARY_1D, ARRAY_2D, ARRAY_3D]
imm2 is a 4-bit size

51.TLD4

Description:

Texture load 4 texels

Usage:

```
TLD4(.AFFI)(.DC)(.NDV)(.T)(.P)(.NODEP)(.mode)(.PTP)(.mode) rd, r1, r2, imm1, dim, imm2;
TLD4.B(.AFFI)(.DC)(.NDV)(.T)(.P)(.NODEP)(.mode)(.PTP)(.mode) rd, r1, r2, 0x0, dim, imm2;
```

Others:

mode is one of [R, G, B, A]
dim is one of [1D, 2D, 3D, ARRARY_1D, ARRAY_2D, ARRAY_3D]
mm2 is a 4-bit size

52.TXQ

Description:

Texture Query

Usage:

`TEX`

Others:

53.TEXDEPBAR

Description:

Texture dependent barriar

Usage:

`DEXDEPBAR imm;`

Others:

54.LDC

Description:

Load from constant

Usage:

```
LDC(.size)(.mode) rd, c[imm1][r1];
LDC(.size)(.mode) rd, c[imm1][imm2];
LDC(.size)(.mode) rd, c[imm1][r1+imm2];
```

Others:

size is one of [U8, S8, U16, S16, 32, 64, 128, U.128]
mode is one of [IL, IS, ISL]

55.LD

Description:

Load from memory

Usage:

```
LD(.E)(.cache)(.size) rd, [r1];
LD(.E)(.cache)(.size) rd, [imm];
LD(.E)(.cache)(.size) rd, [r1+imm];
```

Others:

cache is one of [CS, CG, CV]
size is one of [U8, S8, U16, S16, 32, 64, 128, U.128]

56.LDG

Description:

Non-coherent global memory load

Usage:

`LDG(.size) rd, [r1];`

Others:

size is one of [64, 128]

57.LDL

Description:

Load from local memory

Usage:

```
LDL(.size)(.cache) rd, [r1];
LDL(.size)(.cache) rd, [imm];
LDL(.size)(.cache) rd, [r1+imm];
```

Others:

size is one of [U8, S8, U16, S16, 32, 64, 128, U.128]
cache is one of [CG, LU, CV]

58.LDS

Description:

Load from shared memory

Usage:

```
LDS(.size) rd, [imm];
LDS(.size) rd, [r1];
LDS(.size) rd, [r1+imm];
```

Others:

size is one of [U8, S8, U16, S16, 32, 64, 128, U.128]

59.LDSLK

Description:

Load from shard memory and lock

Usage:

```
LDSLK(.size) p1, rd, [r1];
LDSLK(.size) p1, rd, [imm];
LDSLK(.size) p1, rd, [r1+imm];
```

Others:

size is one of [U8, S8, U16, S16, 32, 64, 128, U.128]

60.ST

Description:

Store to memory

Usage:

```
ST(.E)(.size)(.cache) [rd], r1;
ST(.E)(.size)(.cache) [imm], r1;
ST(.E)(.size)(.cache) [rd+imm], r1;
```

Others:

size is one of [U8, S8, U16, S16, 32, 64, 128, U.128]
cache is one of [CG, CS, WT]

61.STL

Description:

Store to local memory

Usage:

```
STL(.size)(cache) [rd], r1;
STL(.size)(cache) [imm], r1;
STL(.size)(cache) [rd+imm], r1;
```

Others:

size is one of [U8, S8, U16, S16, 32, 64, 128, U.128]
cache is one of [CG, CS, WT]

62.STS

Description:

Store to shared memory

Usage:

```
STS(.size) [rd], r1;
STS(.size) [imm], r1;
STS(.size) [rd+imm], r1;
```

Others:

size is one of [U8, S8, U16, S16, 32, 64, 128, U.128]

63.STSCUL

Description:

Store to shared memory conditionally and unlock

Usage:

```
STSCUL(.size) p1, [rd], r1;
STSCUL(.size) p1, [imm], r1;
STSCUL(.size) p1, [rd + imm], r1;
```

Others:

size is one of [U8, S8, U16, S16, 32, 64, 128, U.128]

64.ATOM

Description:

Atomic memory operation

Usage:

```
ATOM(.E)(.op_type)(.type).FTZ.RN rd, [r1+imm], r2;
ATOM(.E)(.op_type)(.type).FTZ.RN rd, [r1], r2;
ATOM(.E)(.op_type)(.type).FTZ.RN rd, [imm], r2;
ATOM(.E).CAS rd, [r1+imm] r2;
ATOM(.E).CAS rd, [imm] r2;
ATOM(.E).CAS rd, [r1] r2;
```

本文导读目录

点我提问

Others:

op_type is one of [ADD, MIN, MAX, INC, DEC, AND, OR, XOR, EXCH]
type is one of [S32, U63, F32, U128, S64]

65.RED

Description:

Atomic memory reduction operation

Usage:

```
RED(.E)(.op_type)(.type).FTZ.RN [r1+imm], r2;
RED(.E)(.op_type)(.type).FTZ.RN [imm], r2;
RED(.E)(.op_type)(.type).FTZ.RN [r1], r2;
```

Others:

op_type is one of [ADD, MIN, MAX, INC, DEC, AND, OR, XOR, EXCH]
type is one of [S32, U63, F32, U128, S64]

66.CCTL

Description:

Cache control

Usage:

```
CCTL(.E)(.mode1)(.mode2) [r1];
CCTL(.E)(.mode1)(.mode2) [imm];
CCTL(.E)(.mode1)(.mode2) [r1+imm];
```

Others:

mode1 is one of [U, CRS, C, I]
mode2 is one of [PF1, PF1.5, PF2, WB, IV, IVALL, RS, RSLB, WBALL]

67.CCTLL

Description:

Cache control(Local)

Usage:

```
CCTLL(.mode1)(.mode2) [r1];
CCTLL(.mode1)(.mode2) [imm];
CCTLL(.mode1)(.mode2) [r1+imm];
```

Others:

mode1 is one of [U, CRS, C, I]
mode2 is one of [PF1, PF1.5, PF2, WB, IV, IVALL, RS, RSLB, WBALL]

68.MEMBAR

Description:

Memory barrier

Usage:

`MEMBAR(.mode);`

Others:

mode is one of [SYS, GL, CTA]

69.SUCLAMP

Description:

Surace clamp

Usage:

`SUCLAMP(.1D)(.mode)(.U32) p1, rd, r1 composite_operand, imm;`

Others:

mode is one of [SR.R2, SD.R4, SD.R8, SD.R16, P1.R1, P1.R2. P1.R4, P1.R8, P1.R16, B1.R1, B1.R2, B1.R4, B1.R8, B1.R16]

70.SUBFM

Description:

Surface bit field merge

Usage:

```
SUBFM(.3D) p1, rd, r1, composite_operand, r3;
SUBFN(.3D) p1, rd, r2, c[][];
```

Others:

71.SUEAU

Description:

Surface effective address

Usage:

```
SUEAU rd, r1, composite_operand, r3;
SUEAU rd, r1, r2, c[][];
```

Others:

72.SULDGA

Description:

Surface load generic address

Usage:

```
SULDGA.B(.d_type)(.s_type)(.cache)(.mode) rd, [r1], r2, (!)p1;
SULDGA.B(.d_type)(.s_type)(.cache)(.mode) rd, [r1], c[][], (!)p1;
```

Others:

d_type is one of [U8, S8, U16, S16, 32, 64, 128]
s_type is one of [U32, S32, U8, S8]
cache is one of [CA, CS, CV, and default CG]
mode is one of [Z, TRAP, SDCL]

73.SUSTGA

Description:

Surface store generic address

Usage:

Others:

74.BRA

Description:

Branch to relative address

Usage:

`BRA(.compare_op)(.U) target_pc;`

Others:

compare_op is one of [F, LE, EQ, LE, GT, NE, GE, NE, GE, NUM, NAN, LTU, EQU, LEU, GTU, NEU, GEU, OFF, LO, SFF, LS, HI, SFT, HS, OFI
CSM_TR, CSM_MX, TCSM_TA, FCSM_TR, FCSM_TR, RLT, and RGT]

75.BRX

Description:

Branch to relative indexed address

Usage:

```
BRX(.LMT)(.compare_op) r1, imm
BRX(.LMT)(.compare_op) c[imm][r1+imm]
```

Others:

compare_op is one of [F, LE, EQ, LE, GT, NE, GE, NE, GE, NUM, NAN, LTU, EQU, LEU, GTU, NEU, GEU, OFF, LO, SFF, LS, HI, SFT, HS, OFI
CSM_TR, CSM_MX, TCSM_TA, FCSM_TR, FCSM_TR, RLT, and RGT]

76.JMP

Description:

Jump to absolute address

Usage:

```
JMP(.U)(.LMT)(.compare_op) imm;
JMP(.U)(.LMT)(.compare_op) c[imm][imm];
```

Others:

compare_op is one of [F, LE, EQ, LE, GT, NE, GE, NE, GE, NUM, NAN, LTU, EQU, LEU, GTU, NEU, GEU, OFF, LO, SFF, LS, HI, SFT, H
CSM_TR, CSM_MX, TCSM_TA, FCSM_TR, FCSM_TR, RLT, and RGT]

本文导读目录

点我提问

77.JMX

Description:

Jump to absolute indexed address

Usage:

```
JMX(.LMT)(.compare_op) r1, imm;
JMX(.LMT)(.compare_op) c[imm][r1+imm];
```

Others:

78.CAL

Description:

Call to relative address

Usage:

```
CAL(.NOINC) imm;
CAL(.NOINC) c[imm][imm];
```

Others:

79.JCAL

Description:

Call to absolute address

Usage:

`JCAL(.NOINC) imm;`

Others:

80.RET

Description:

Return from call

Usage:

`RET(.compare_op);`

Others:

compare_op is one of [F, LE, EQ, LE, GT, NE, GE, NE, GE, NUM, NAN, LTU, EQU, LEU, GTU, NEU, GEU, OFF, LO, SFF, LS, HI, SFT, HS, OFI
CSM_TR, CSM_MX, TCSM_TA, FCSM_TR, FCSM_TR, RLT, and RGT]

81.BRK

Description:

Break from loop

Usage:

`BRK(.compare_op);`

Others:

compare_op is one of [F, LE, EQ, LE, GT, NE, GE, NE, GE, NUM, NAN, LTU, EQU, LEU, GTU, NEU, GEU, OFF, LO, SFF, LS, HI, SFT, HS, OFI
CSM_TR, CSM_MX, TCSM_TA, FCSM_TR, FCSM_TR, RLT, and RGT]

82.CONT

Description:

Continue in loop

Usage:

`CONT(.compare_op);`

Others:

compare_op is one of [F, LE, EQ, LE, GT, NE, GE, NE, GE, NUM, NAN, LTU, EQU, LEU, GTU, NEU, GEU, OFF, LO, SFF, LS, HI, SFT, H
CSM_TR, CSM_MX, TCSM_TA, FCSM_TR, FCSM_TR, RLT, and RGT]

83.SSY

Description:

Set sync relative address

Usage:

```
SSY imm;
SSY c[][];
```

Others:

84.PBK

Description:

Pre-break relative address

Usage:

```
PBK imm;
PBK c[][];
```

Others:

compare_op is one of [F, LE, EQ, LE, GT, NE, GE, NE, GE, NUM, NAN, LTU, EQU, LEU, GTU, NEU, GEU, OFF, LO, SFF, LS, HI, SFT, HS, OFI
CSM_TR, CSM_MX, TCSM_TA, FCSM_TR, FCSM_TR, RLT, and RGT]

85.PCNT

Description:

Pre-continue relative address

Usage:

```
PCNT(.NOINC) imm;
PCNT(.NOINC) c[][];
```

Others:

86.PRET

Description:

Pre-return relative address

Usage:

```
PRET(.NOINC) imm;
PRET(.NOINC) c[][];
```

Others:

87.BPT

Description:

Breakpoint/Trap

Usage:

`BPT(.mode) imm;`

Others:

mode is one of [CAL, PAUSE, TRAP, INT, DRAIN]

本文导读目录

点我提问

88.EXIT

Description:

Exit program

Usage:

`EXIT;`

Others:

89.NOP

Description:

No operation

Usage:

`NOP;`

Others:

90.S2R

Description:

Special register to register

Usage:

`S2R rd, sp;`

Others:

sp includes [SR_TID.x, SR_TID.y, SR_TID.z, SR_CTAID.x, SR_CTAID.y, SR_CTAID.z, SR_LANEID, SR_VIRTID, SR_PM0-SMPM7, SR_CLOCI
SR_CLOCKHI, SR_GTMASK, SR_GEMASK, SR_LTMASK, SR_LEMASK, SR_EQMASK, SR_VIRTCFG, SR_GLOBALTIMELO, SR_GLOBALTI
SR_SMEMSZ]

91.B2R

Description:

Barrier to register

Usage:

`B2R(.mode) rd, imm;`

Others:

mode is one of [RESULT, WARP]

92.BAR

Description:

Barrier synchronization

Usage:

```
BAR(.mode) imm;
BAR(.mode) imm1, imm2;
```

Others:

mode is one of [SYNC, ARR, RED.POPC, SCAN, SYNCALL]

93.VORE

Description:

Query condition across threads

Usage:

```
VOTE(.mode1) rd, p1, (!)p2;
VOTE.VTG imm;
```

Others:

mode1 is one of [ALL, ANY, and EQ]
mode2 is one of [R, A, and RA]

94.ISUB

Description:

Integer subtrack

Usage:

`ISUB(.SAT)(.X)(.P0) rd(.CC), (-r1), (-)composite_operand;`

Others:

成功客户案例分享

公告
信任中心
举报中心

产品图标

云教程
开发者交流
提交建议

阿里云微信
阿里云微博
阿里云客户满意中心

热门产品	9.9元云服务器	云数据库RDS	云存储OSS	NAT网关	负载均衡	域名注册	网站建设	大数据
	云计算							
用户热搜	网站备案	网安法	CDN加速	API网关	企业邮箱	whois查询	视频直播	视频转码
	云安全							
更多推荐	数据科研社区	阿里云大学	学生机	IT论坛	数据可视化	云虚机	com域名	cn域名
	合规安全解决方案							

关于我们 法律声明 廉正举报 友情链接

阿里巴巴集团 淘宝网 天猫 聚划算 全球速卖通 阿里巴巴国际交易市场 1688 阿里妈妈 飞猪 阿里云计算 YunOS 阿里通信 万网 高德 UC 友盟 虾米 阿里星球 来往 钉钉 支付宝

本文导读目录

点我提问