Банки общей памяти

Банки общей памяти

- Для увеличения полосы пропускания устройство, на котором расположена общая память, разделено на подмодули («**банки**»)
  - **n** – число банков
  - **m** – сколько последовательных байтов может отдать каждый банк за цикл
- Адресное пространство общей памяти разделено на **n** непересекающихся подмножеств, расположенных в разных банках
- Банки работают независимо друг-от-друга и могут вместе выдать максимум **n*m** байтов за один цикл

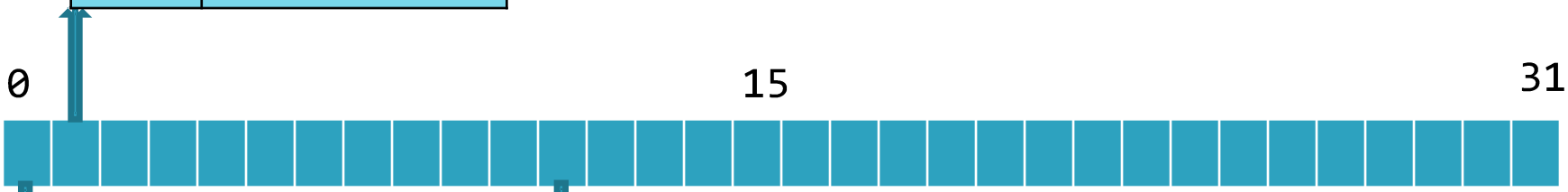
Банки общей памяти на Fermi

- 32 банка, каждый банк может выдать за 2 такта ядер **одно** 32-битное слово (4 последовательных байта)
- Последовательные 32-битные слова располагаются в последовательных банках
  - Номер банка для слова по адресу addr: `(addr / 4) % 32`
- За два такта ядер общая память может отдать 128 байт

Банк 1

4	
132	
260	
388	
516	
…	

Банк 0	Банк 1	Банк 2	Банк 3	…
0	4	8	12	…
128	132	136	140	…
256	260	264	268	…
…	…	…	…	…



Банк 0

0	
128	
256	
384	
512	
…	

Банк 11

44	
172	
300	
428	
556	
…	

Обращения в общую память

- Обращение выполняется одновременно всеми нитями варпа (SIMT)
- Банки работаю параллельно
  - Если варпу нитей нужно получить 32 4-байтных слова, расположенных в разных банках, то такой запрос будет выполнен одновременно всеми банками
    - Каждый банк выдаст соответствующее слово
    - Пропускная способность = **32 х пропускная способность банка**
- Поддерживается рассылка (broadcast):
  - Если часть нитей (или все) обращаются к одному и тому же 4-х байтному слову, то нужное слово будет считано из банка и роздано соответствующим нитям (broadcast) без накладных расходов

Банк конфликты

- Если хотя бы два нужных варпу слова расположены в одном банке, то такая ситуация называется «**банк конфликтом**» и обращение в память будет «**сериализовано**»:
  - Такое обращение аппаратно разбивается на серию обращений, не содержащих банк конфликтов
    - Если число обращений, на которое разбит исходный запрос, равно n, то такая ситуация называется **банк-конфликтом порядка n**
    - Пропускная способность при этом падает в n раз

Банки общей памяти на Kepler

- 32 банка, каждый банк может выдать за 1 такт ядер 8 байтов
  - На Kepler частота ядер в 2 раза меньше, чем на Fermi
- Два режима разбиения общей памяти на банки:
  - Последовательные 32-битные слова располагаются в последовательных банках: `(addr / 4) % 32`
  - Последовательные 64-битные слова располагаются в последовательных банках: `(addr / 8) % 32`
- За два такта ядер общая память может отдать 256 байт

Примеры банк-конфликтов

```
extern __shared__ float char[];
float data = shared[BaseIndex + s * threadIx.x];// конфликты
                                                зависят от s
```

- Нити `threadIx.x` и `(threadIx.x + n)` обращаются к элементам из одного и того же банка когда s*n делится на 32 (число банков).
  - S = 1 :

    `shared[BaseIndex + threadIx.x]` // нет конфликта
  - S = 2 :

    `shared[BaseIndex + 2*threadIx.x]` // конфликт 2-го порядка

    Например, между нитями `threadIx.x=0` и `(threadIx.x = 16)` – попадают в один варп!

Распространенная проблема

- Пусть в общей памяти выделена плоская плотная матрица шириной, кратной 32, и соседние нити варпа обращаются к соседним элементам столбца

```
__shared__ int matrix[32][32]
matrix[thredIdx.x][4] = 0;
```

Распространенная проблема

- Пусть в общей памяти выделена плоская плотная матрица шириной, кратной 32, и соседние нити варпа обращаются к соседним элементам столбца

```
__shared__ int matrix[32][32]
matrix[thredIdx.x][4] = 0;
```

Банк конфликт 32-го порядка

Распространенная проблема

```
__shared__ int matrix[32][32]
  matrix[thredIdx.x][4] = 0;
```

Банк конфликт 32-го порядка 

Решение: набивка

```
__shared__ int matrix[32][32 + 1]
matrix[thredIdx.x][4] = 0; //нет конфликта
```

Распространенная проблема

Пусть банков 10, матрица 10x10