Georgia Tech

Harmonica GPU

Georgia Tech

Objectives

- Detailed look at the implementation of a SIMT GPU
- Example of the type of information propagated down the pipeline
- Basis for the next assignment and the default project

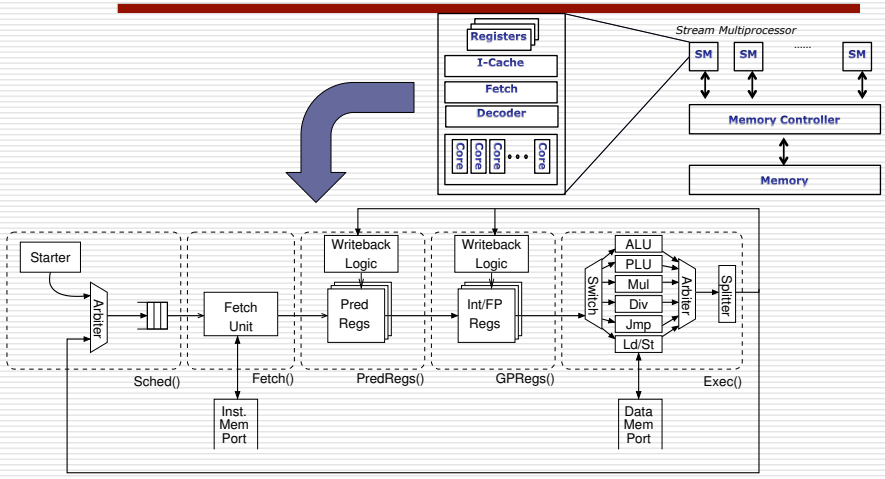
Reading

- C. Kersey, “HARP Instruction Set Manual”
- CHDL Architecture Description
  - https://github.com/cdkersey/harmonica2
  - Note that this is under active development so the current code base/definitions may be changing

The Harmonica GPU

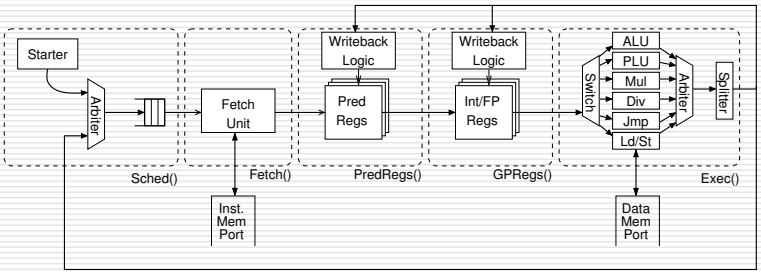
- An instance of the HARP family of ISAs
  - SIMT oriented RISC-like ISAs
- Parametric SIMT GPU
  - Datapath width, instruction encodings, #registers are configurable
- Lightweight, simplified pipeline with multi-warp execution
  - No thread blocks
  - Nested parallelism
- Designed as a memory side or CPU accelerator for data intensive computing

System Organization



- Parametric architecture
  - Key Parameters - #SMs, #lanes/SM, #registers, and data-widths.

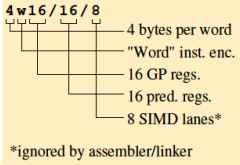
Microarchitecture



- Customizable, multithreaded, SIMT soft core
  - Generated from an architecture specification
- Supported by a generated HARP Tool assembler/ linker/emulator
- Small - ~1500 lines of C++

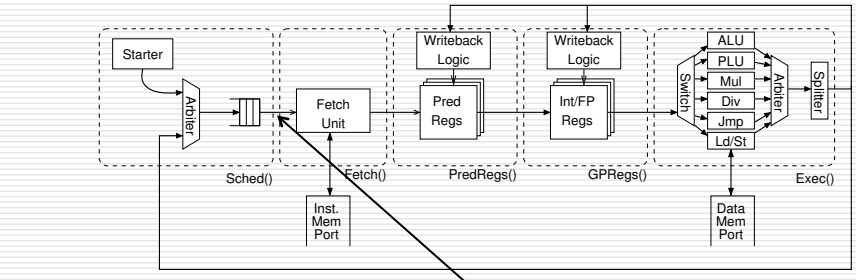
```
/* Return in r0 dot product of vectors
   of real values pointed to by r0 and
   r1, length in r2 */
dotprod: ldi %r3, #0;
dploop:  ld %r4, %r0, #0;
         ld %r6, %r1, #0;
         subi %r2, %r2, #1;
         addi %r0, %r0, __WORD;
         addi %r1, %r1, __WORD;
         rtop @p0, %r2;
         fmul %r4, %r4, %r6;
         fadd %r3, %r3, %r4;
   @p0 ? jmpi dploop;
         ori %r0, %r3, #0;
         jmpr %r5;
```

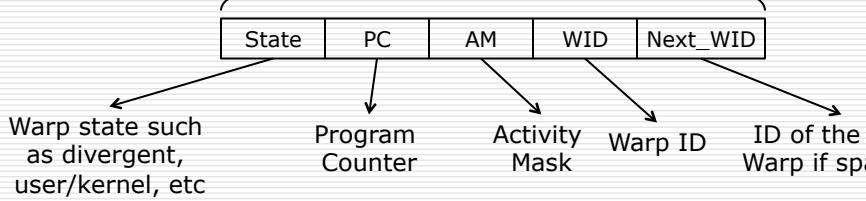
ArchID String



*ignored by assembler/linker

Microarchitecture: Schedule





Warp state such as divergent, user/kernel, etc

Program Counter

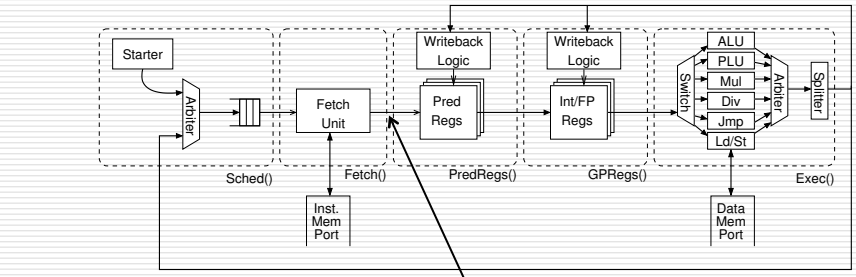
Activity Mask

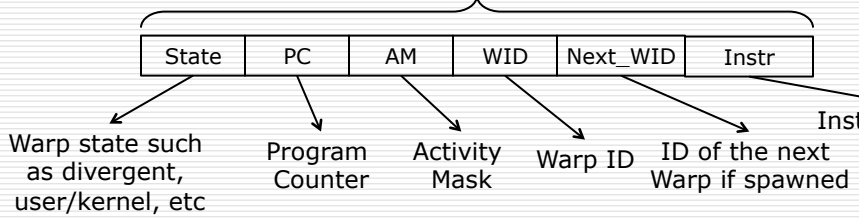
Warp ID

ID of the next Warp if spawned

(7)

Microarchitecture: Fetch





Warp state such as divergent, user/kernel, etc

Program Counter

Activity Mask

Warp ID

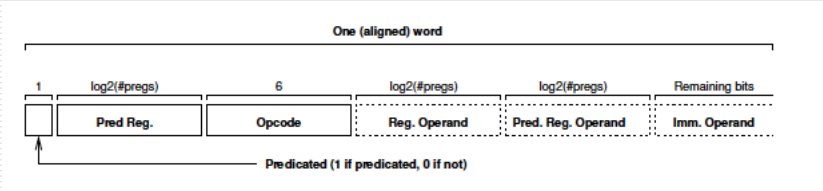
ID of the next Warp if spawned

Instruction

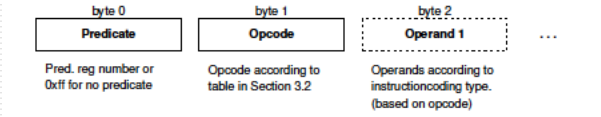
(8)

Harmonica Instruction Format

Word Encoding

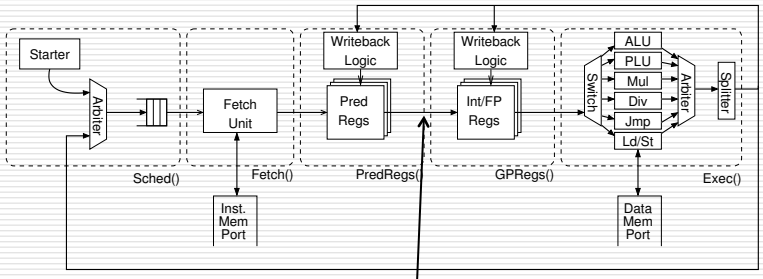


Byte Encoding

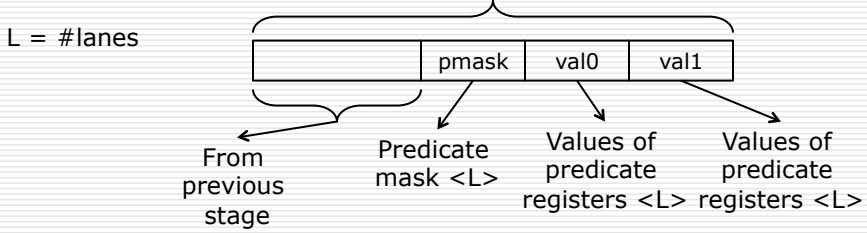


(9)

Microarchitecture: PRF Access

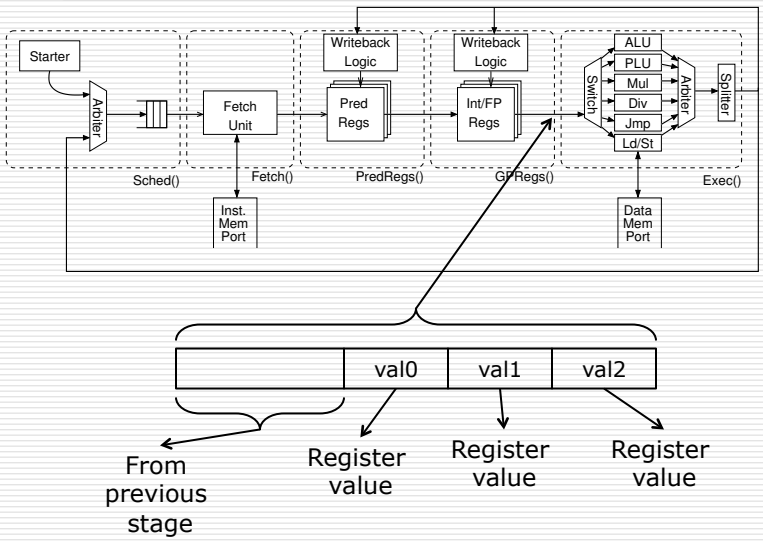


L = #lanes



(10)

Microarchitecture: GPR Access



From previous stage

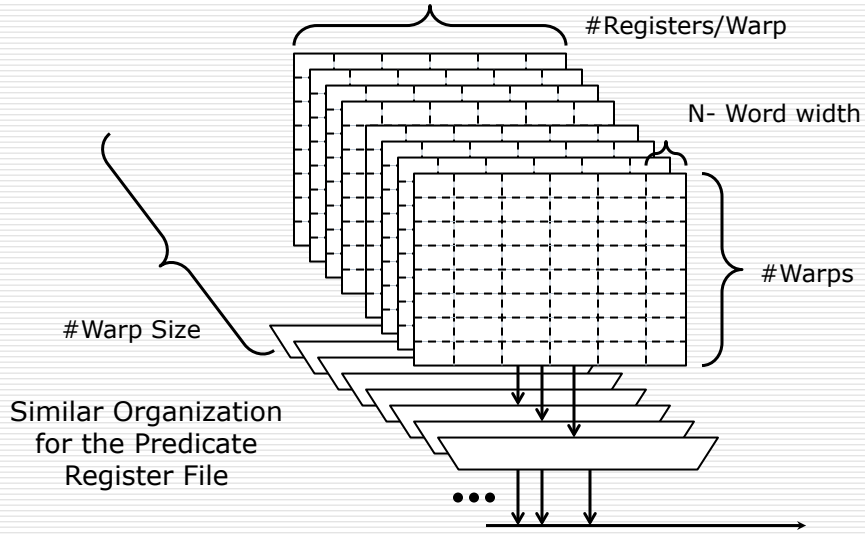
Register value

Register value

Register value

(11)

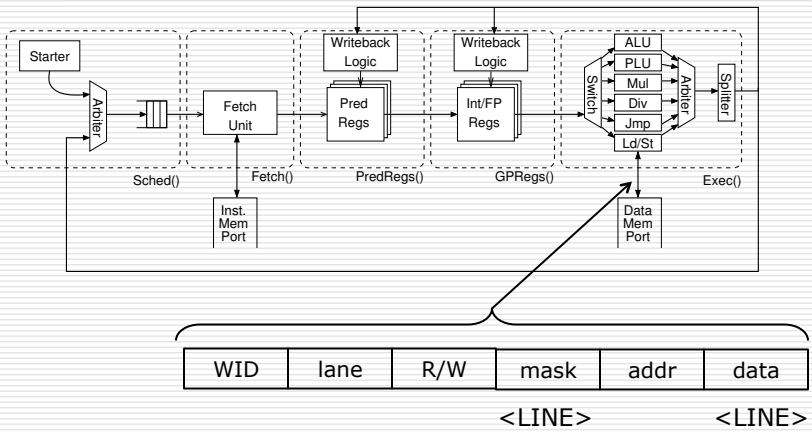
Register File Organization



Similar Organization for the Predicate Register File

(12)

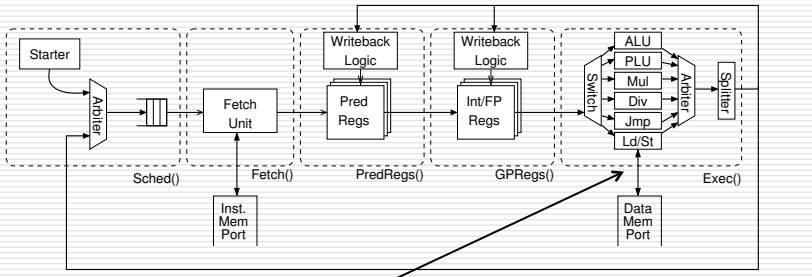
Microarchitecture: Cache Request

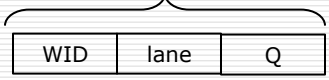


<LINE> <LINE>

(13)

Microarchitecture: Cache Response



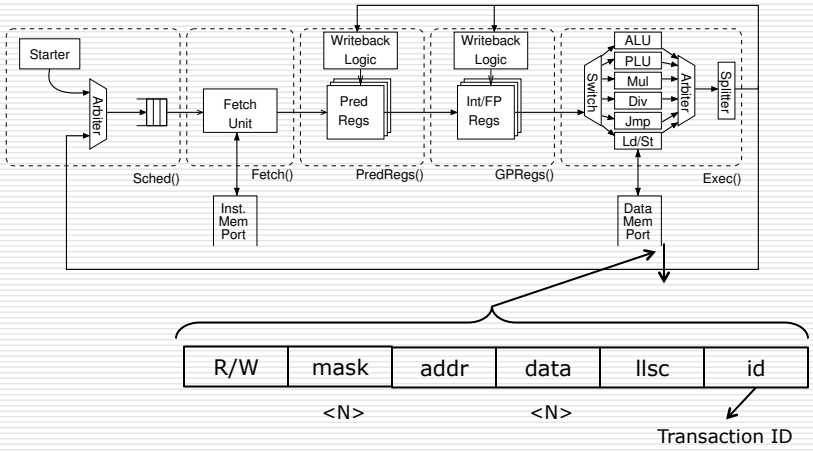


<LINE>

- Q – cache line number of words

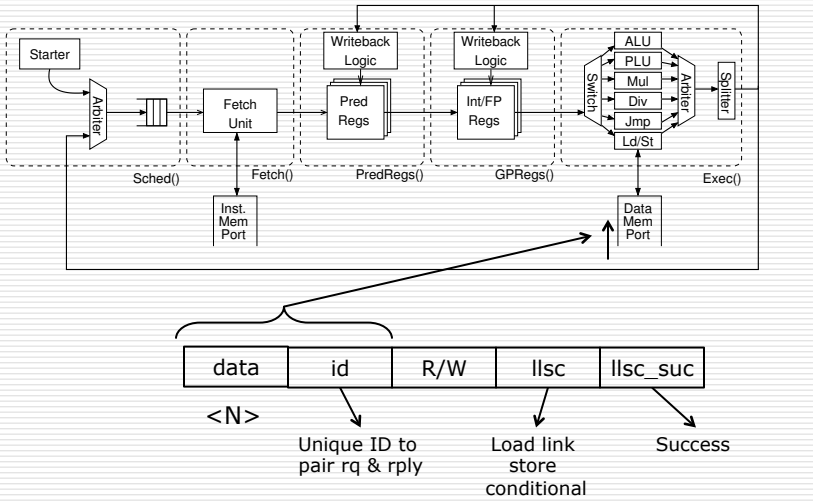
(14)

Microarchitecture: Memory Request

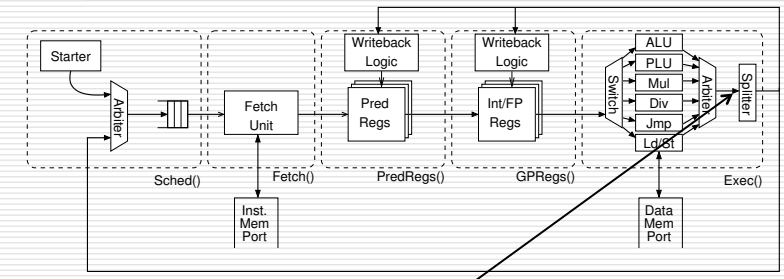


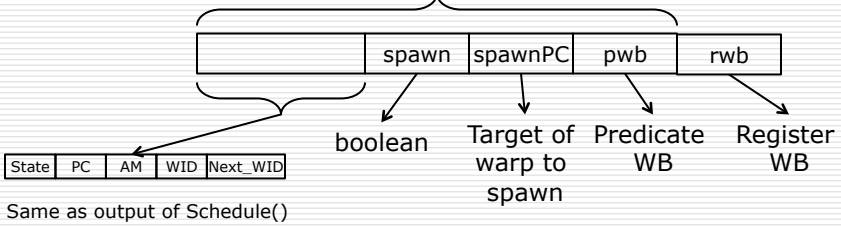
Fetch N B-bit bytes from location addr (word addressed)

Microarchitecture: Memory Response



Microarchitecture: FU Out

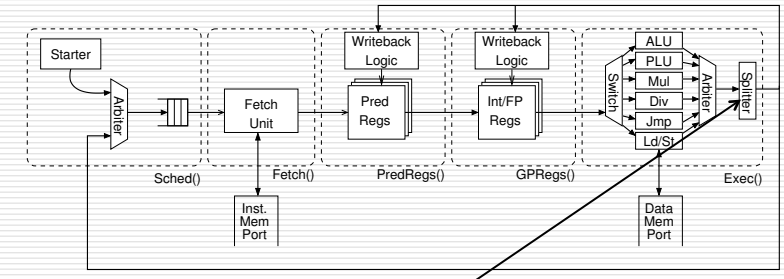


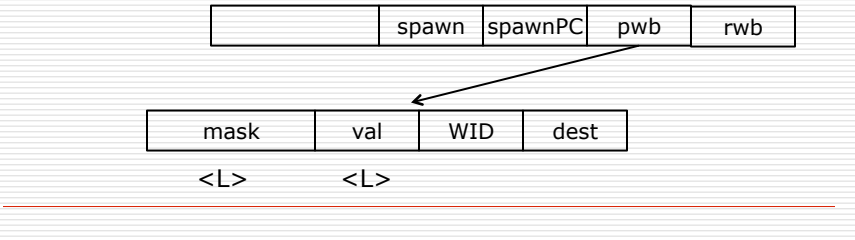


Same as output of Schedule()

(17)

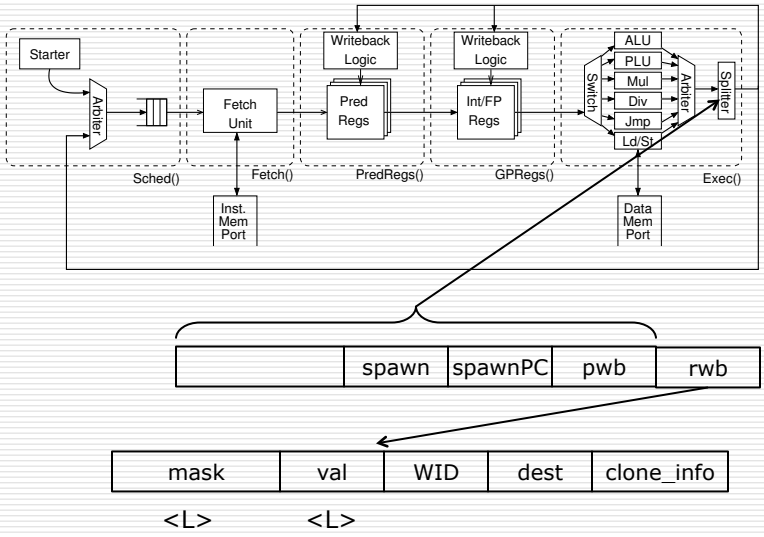
Microarchitecture: FU Out (2)



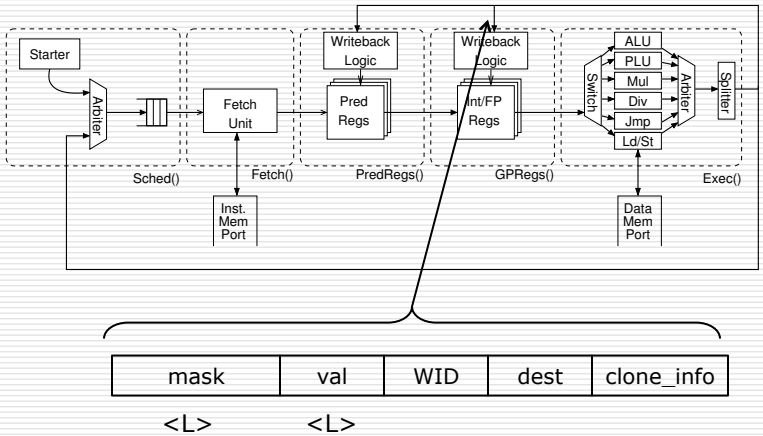


(18)

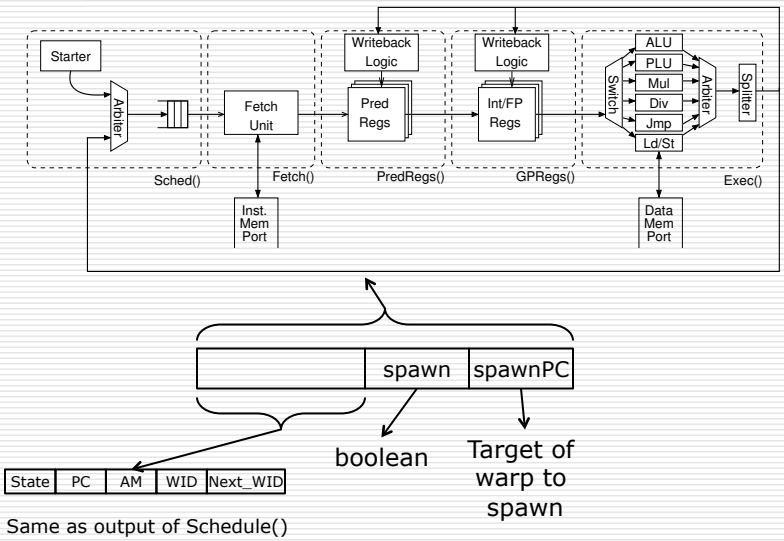
Microarchitecture: FU Out (3)



Microarchitecture: GPR Writeback

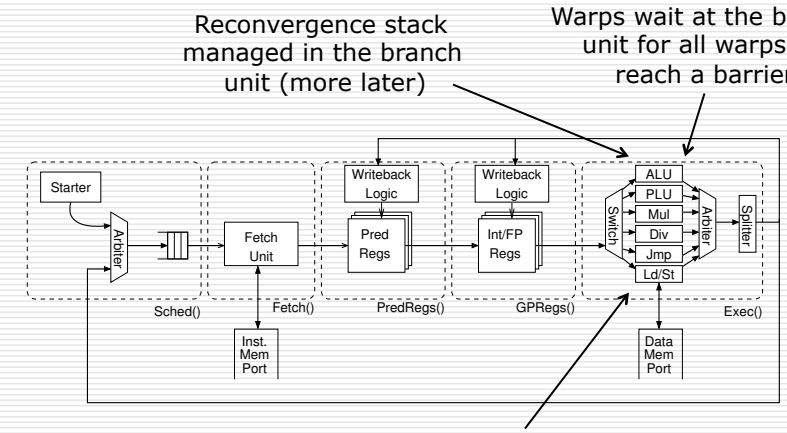


Microarchitecture: Next Instruction



Same as output of Schedule()

Microarchitecture: Stalled Warps



Some Miscellaneous Observations

- Recirculating warp model of execution
  - Waiting warps distributed through the data path
    - Barriers, divergent warps, memory accesses
- Warps can spawn other warps → nested parallelism
- Elementary scheduling, and fetch policies
- Supports kernel mode execution and interrupts (lane 0)

(23)