ECE 4750 Computer Architecture, Fall 2015

T15 Advanced Processors: Multithreaded Processors

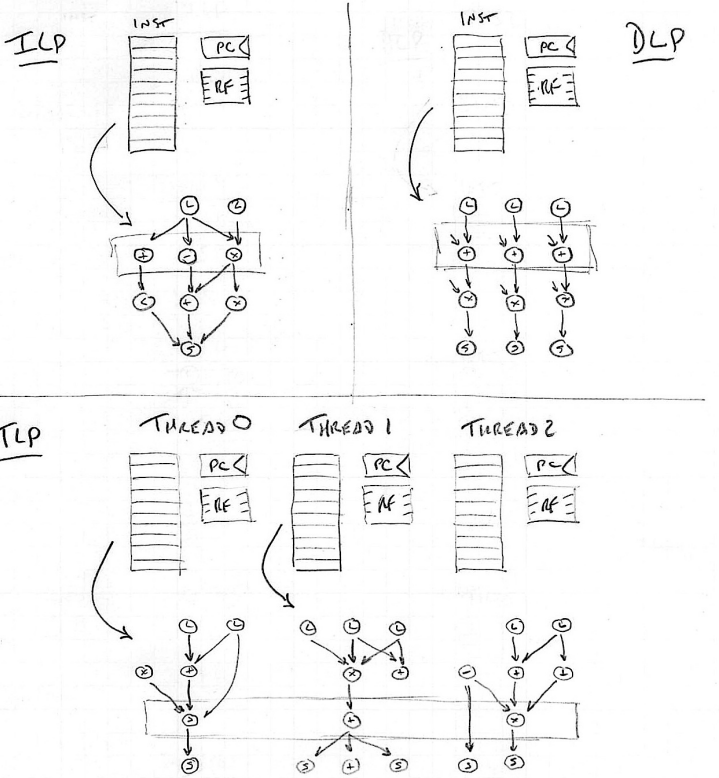
School of Electrical and Computer Engineering
Cornell University

revision: 2015-11-20-12-10

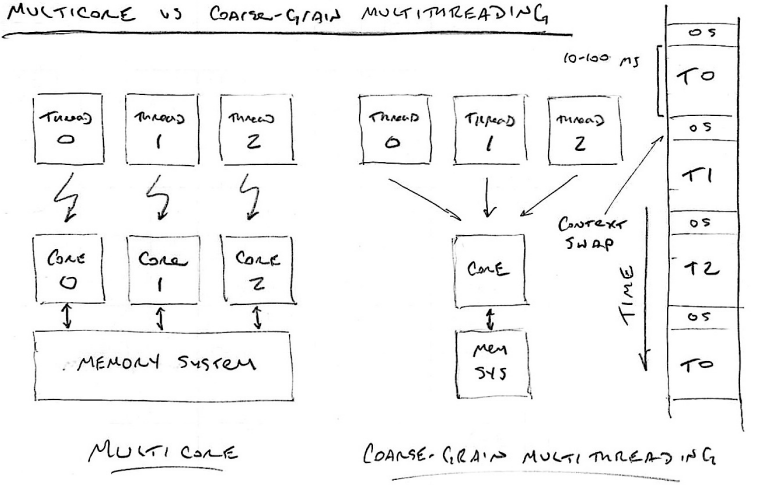
ILP vs DLP vs TLP

ILP = INSTRUCTION-LEVEL PARALLELISM
DLP = DATA-LEVEL PARALLELISM
TLP = THREAD-LEVEL PARALLELISM

SISD	MISD
SIMD	MIMD



- TLP from multiprogramming (multiple applications)
- TLP from multithreaded applications
  - ↳ run one application faster with multiple threads
  - ↳ pthreads, CILK, TBB, OpenMP



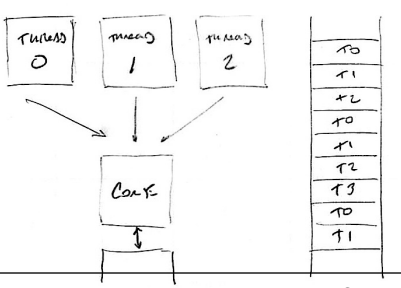
FINE-GRAIN MULTITHREADING

HARDWARE SUPPORT TO ENABLE INTERLEAVING MULTIPLE THREADS ON A SINGLE CORE AT A VERY FINE GRANULARITY.

WE WILL DISCUSS TWO VARIANTS OF FINE-GRAIN MULTITHREADING:

- VERTICAL MULTITHREADING
- SIMULTANEOUS MULTITHREADING (SMT)

VERTICAL MULTITHREADING



SWITCH BETWEEN THREADS AT A CYCLE-BY-CYCLE GRANULARITY

STATE FOR ALL THREE THREADS KEPT IN DEDICATED HARDWARE

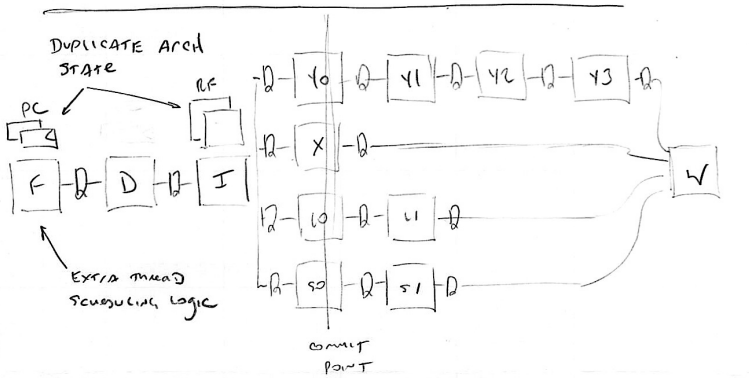
THREAD SCHEDULING HANDLED BY HARDWARE

MT CODE EXAMPLE

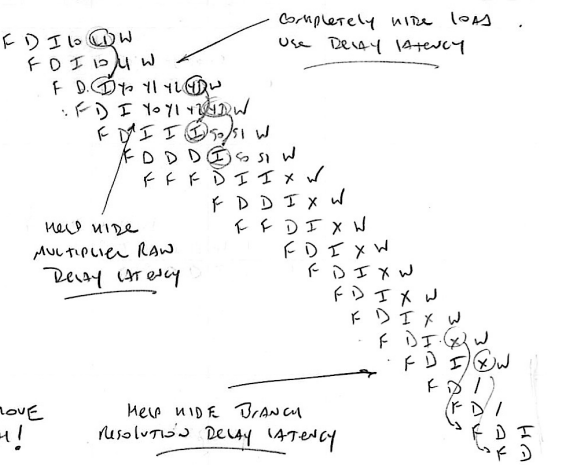
```
j = thread_id;
start = j * (n / nthreads);
for (int i = start; i < n/nthreads; i++)
    B[i] = A[i] * C;
```

ASSUME n multiple of nthreads

VERTICAL MULTITHREADING MICROARCHITECTURE

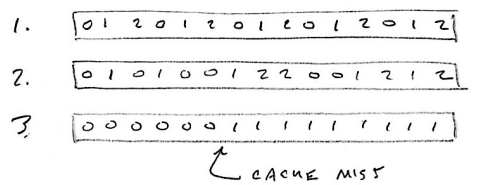



```
T0: LW r1,0(r2)
T1: LW r1,0(r2)
T0: MUL r3,r1,r4
T1: MUL r3,r1,r4
T0: SW r3,0(r5)
T1: SW r3,0(r5)
T0: ADDW r2,r2,4
T1: ADDW r2,r2,4
T0: ADDW r5,r5,4
T1: ADDW r5,r5,4
T0: ADDW r7,r7,-1
T1: ADDW r7,r7,-1
T0: bgtz r7,loop
T1: bgtz r7,loop
```

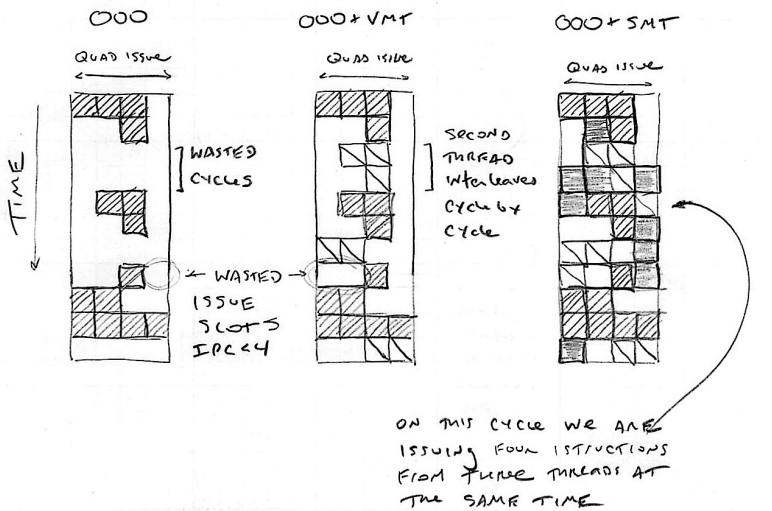


SCHEDULING POLICIES

1. STATIC FIXED INTERLEAVING
   - EACH OF N THREADS EXECUTE ONE INSTRUCTION EVERY N CYCLES
   - IF THREAD IS NOT READY TO GO CAN EITHER:
     - STALL ENTIRE FRONT-END
     - INSERT BUBBLE, BUT DO NOT STALL FRONT-END
   - CAN POTENTIALLY ELIMINATE INTERLOCKING + BYPASS NETWORK
2. DYNAMIC INTERLEAVING
   - HARDWARE KEEPS TRACK OF WHICH THREADS ARE READY
   - PICKS NEXT THREAD TO EXECUTE BASED ON PRIORITY SCHEME
3. COARSE-GRAIN HARDWARE INTERLEAVING
   - USE THREADS TO HIDE OCCASIONAL CACHE MISS LATENCY



Simultaneous Multithreading (SMT)

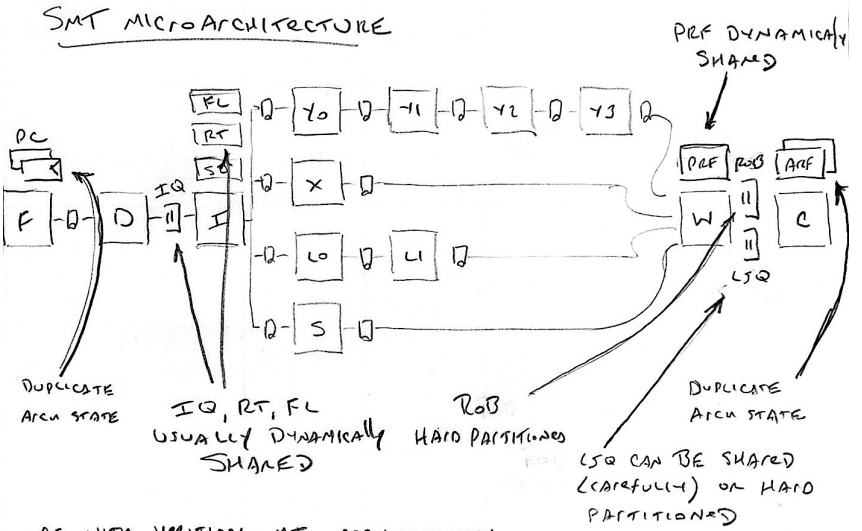


SMT uses the fine grain control already present in an OOO superscalar processor to allow instructions from different threads to issue at the same time

Add multiple Fetch engines to enable fetching + decoding instructions from different threads

IQ does not know about threads. Simply finds instructions that are ready to issue — these instructions may or may not be from different threads

* SMT adapts to parallelism type
  - For applications with High ILP but no TLP, app can use entire width of the machine
  - For applications with High TLP but less ILP, the width of the machine is shared across threads



AS WITH VERTICAL MT, ARCHITECTURAL STATE MUST BE DUPLICATED

MICROARCHITECTURAL STATE CAN EITHER BE

- DUPLICATED AT DESIGN TIME
- HARD PARTITIONED AT BOOT TIME
- DYNAMICALLY SHARED AT EXECUTION TIME

USUALLY NEED TO INCREASE SIZE OF SHARED DATA STRUCTURES

IQ, PRF, LSQ

THREAD SCHEDULING

FETCH FROM THREAD WITH THE LEAST INSTRUCTIONS IN FLIGHT

Draw a pipeline diagram for the assembly loop to the right executing on a dual-issue IO2L microarchitecture with register renaming, memory disambiguation, perfect branch prediction, and *two SMT threads*. Draw the diagram to illustrate how both threads simultaneously execute the first iteration of the loop.

```
lw    r1, 0(r2)
mul   r3, r1, r4
sw    r3, 0(r5)
addiu r2, r2, 4
addiu r5, r5, 4
addiu r7, r7, -1
bgtz  r7, loop
```