3 INSTRUCTION LEVEL PARALLELISM

i way - instruction pipelining

multiple instructions are overlapped in execution.

each stage in the pipeline is referred to as pipe stage or pipe segment.

each stage must be able to complete its operation in a clock cycle. try to balance the time taken by each stage.

the length of a processor cycle is determined by the slowest stage. so pipeline designer's goal is to balance the length of each pipeline stage.

time / instruction

$$= \frac{\text{time for exec. in unpipelined stage}}{\text{no. of pipeline stages.}}$$

cores like organs.

CS 252 2011 & 2012 PPTs.

ACM 47 (October 10) 71-75 2004

latency lags bandwidth.

classmate
Date 22·1·2020
Page 1

CPI (cycles per inst.) for a pipelined processor, is given by the base CPI

CPI = sum of base CPI and all contributions from stalls.

pipeline CPI = ideal pipeline CPI + structural stalls + (data inst.)
data hazard stalls + control stalls.
(write after read...) (branch)

THE BASICS of THE RISCV INSTRUCTION SET

its concerned with load and store operation. Computations all operations on data apply to data in register. the only operations that affect memory are the load & store operations.

1. instruction fetch cycle. (IF)
2. instruction decode and register fetch (ID)
3. execution / effective address cycle (EX)
4. memory access. (MEM)
5. write back cycle. (WB).

branch instructions	3 cycles
Store instruction	4 cycles
other instructions	5 cycles.

pipelining increases the (throughput) of processor instruction. usually slight increase in the execution time of

each instruction. major hurdle of pipelining - Pipeline hazards.

① Structural hazards.

they arise from resource conflicts, when the hardware may not support all possible combinations of instructions simultaneously in overlapped execution.

② data hazard,

when an instruction depends on the result of a previous instruction, in a way that is exposed by the overlapping instruction in the pipeline.

(delay can be reduced by forwarding)

③ control hazards.

may arise from the pipeline of branch and other instructions that change the PC.

(dynamic / static) ← can be avoided by.

↳ by compiler

- loop unrolling
- reordering instructions.

2 normally seperable approaches to exploits ILP.

1- an approach that relies on hardware to help discover and exploit parallelism dynamically.

2- an approach that relies on software technology to find parallelism statically at compile time.

we have to exploit ILP across multiple blocks.

dynamic scheduling.
- hw will change order of exec. while maintaining correctness of the program.

imprecise exceptions must be avoided.

dynamic scheduling example.

fdiv.d f0, f2, f4
fadd.d f10, f0, f8
fsub.d f12, f8, f14 ← can be executed in advance.

to check for these hazards, the ID phase
we must separate the issue process into two parts.
(1) checking for any structural hazards and waiting for the absence of data hazard.

fdiv.d f0, f2, f4
fmul.d f6, f0, f8
faddd.d f0, f10, f14

} write after read hazard.

register renaming 152

study from book for pipelining.
appendix C in 5th/6th Edition.
TLP thread level parallelism.

branch predictor
- global
- local.

format
predictor.