# 1. MECHANICS

## 1.1. Homework

1.1.1. #2 due today by 3pm, solutions out tomorrow morning

1.1.2. #3 handed out

## 1.2. Midterm: no calculators, no assembly code

# 2. MEMORY OPERATIONS

## 2.1. From thursday's lecture

# 3. EXAMPLE PROCESSORS

## 3.1. Six examples of advanced processors so far

3.1.1. Smith & Weiss: PowerPC 601, Alpha 21064

3.1.2. Smith & Sohi: MIPS R10000, AMD K5, Alpha 21164

3.1.3. MPR articles: Alpha 21164, Alpha 21264

## 3.2. PPC 601

3.2.1. unified first level cache requires large block fetches of instructions

3.2.2. compiler predicts branches, at most 2 cycles mispredict penalty

3.2.3. branches & floating point may issue out-of-order compared to integer unit

3.2.4. 3 specialized pipelines (branch, fixed point, floating point)

3.2.5. allows imprecise floating point exceptions, which can be avoided by setting the processor in sequential mode or by inserting test code

## 3.3. Alpha 21064

3.3.1. dynamic 1-bit branch prediction, 4 cycles mispredict penalty

3.3.2. 3 specialized pipelines (fixed-point, floating-point, load/store)

3.3.3. two instructions may issue if they are aligned correctly and use separate pipelines (even two integer ops if one is load/store and other is operation)

3.3.4. allows imprecise floating point exceptions, which can be avoided with a barrier instruction (which guarantees that all preceeding instructions have completed before allowing execution beyond barrier)

## 3.4. R10000

3.4.1. an aggressive superscalar

3.4.2. separate 32K, 2-way first level caches

3.4.3. branch prediction in cache, 2-bit dynamic

3.4.4. on branch prediction, snapshot of register file is saved; up to 4 predictions can be outstanding

3.4.5. register renaming with 64 physical registers

3.4.6. up to 4 instr/cycle dispatch to 3 queues: FP (feeding FP mult & FP add pipelines), integer (feeding 2 partially specialized ALUs) and load/store (feeding one load/store pipeline)

3.4.7. FP and integer queues can issue out-of-order (but they are not quite reservation stations)

3.4.8. ROB is used for precise interrupts and freeing of physical registers, up to 4 instr/cycle retired

### 3.5. 21164

3.5.1. simpler than R10000

3.5.2. separate 8K, direct-mapped first level caches plus 96K combined, 3-way second level on chip

3.5.3. branch prediction in cache, 2-bit dynamic

3.5.4. only one outstanding prediction at a time

3.5.5. 4 instructions fetched/cycle into 1 of 2 buffers for dispatch, each buffer must clear before the next is used

3.5.6. 4 pipelines: 2 integer, FP adder, FP mult

3.5.7. instructions must issue and commit in-order

3.5.8. FP may commit out-of-order and be imprecise

### 3.6. K5

3.6.1. implements complex, variable length x86 ISA

3.6.2. separate 16K instr & 8K data caches

3.6.3. complex predecode (going into cache) breaks up instructions

3.6.4. branch prediction in icache, 1-bit dynamic, BTAC integrated (but only to other cache addresses), only one outstanding

3.6.5. complicated decode into up to 4 ROPs (RISC-like Operations) per cycle (possibly generated from microcode ROM)

3.6.6. ROB handles register renaming, precise interrupts & mispredicted branches

3.6.7. 4 ROPs/cycle dispatched to reservation stations, 4 ROPs/cycle issued to pipelines

3.6.8. 6 pipelines (most with 2 reservation stations): FPU, branch, 2 ALUs, 2 load/store

3.6.9. will examine more later

### 3.7. 21264

3.7.1. very complex implementation

3.7.2. separate 64K, 2-way first level caches, but no second level cache on-chip and first level costs 2 cycles

3.7.3. separate dynamic local and global history predictions are compared for accuracy (we have not examined this scheme)

3.7.4. BTAC integrated with cache (but only to other cache addresses)

3.7.5. register renaming with 80 physical registers

3.7.6. 4 instr/cycle into integer or FP queues (which are basically reservation stations)

3.7.7. 6 instr/cycle out-of-order into pipelines: 2 full ALUs, 2 address ALUs (simple ALU ops), 1 FP add, 1 FP mult

3.7.8. 2 copies of integer register file (one for each cluster: a full ALU + addr ALU) allow each copy to only have 4 read and 6 write ports, but related instructions should go to same cluster

# 4. TOPICS FOR THE MIDTERM

## 4.1. price-performance

4.1.1. measuring performance
4.1.2. price * performance (measured in $-time), lower the better
4.1.3. speed-up & amdahl's law (of decreasing returns)

## 4.2. ISA

4.2.1. definitions: orthogonal, latency vs throughput, static vs dynamic
4.2.2. types of architectures (stack, accumulator, GPR)
4.2.3. types of GPR (memory-memory, register-memory, load-store)
4.2.4. types & locations of operands
4.2.5. addressing
4.2.6. instruction encoding & length
4.2.7. control instructions
4.2.8. ISA & compilers
4.2.9. example ISAs: VAX and MIPS / DLX
4.2.10. comparing RISC and CISC (Bhandarkar & Clark)
4.2.11. comparing types of RISCs (Smith & Weiss)

## 4.3. Implementation

4.3.1. microinstructions
4.3.2. instruction-level parallelism & dependence information (Fisher & Rau)
4.3.3. conflict between sequential execution assumption and pipelining, parallelism & reordering
4.3.4. five parts of advanced pipeline (fetch, dispatch, issue, execution, commit) (Smith & Sohi)
4.3.5. hazards: structural, control, data (3 types), interrupts
4.3.6. resolving hazards in simple pipeline
4.3.7. resolving hazards in advanced pipeline
4.3.8. branch prediction (MPR Gwennap)
4.3.9. register renaming
4.3.10. reorder buffer (Smith & Pleszkun)
4.3.11. reservation stations
4.3.12. load/store address buffers
4.3.13. example implementations