Chapter 4 Review Alex Lundin

---Multiplexor – selects between sources ---Combinational element – operational element such as AND gate or ALU ---State element – a memory element such as a register ---Clocking methodology – way to determine if data is valid and stable relative to the clock ---Edge triggered clocking – state changes occur on a clock edge ---Control signal – used for selection with a multiplexor or directing the operation of a functional unit ---Asserted – signal logically high or true ---Data signal – contains information that is operated on by functional unit ---Deassarted – signal is logically low or false ---Datapath element – unit used to operate on or hold data. Instruction, data memories, the register file, the ALU and adders ---Program counter – register containing address of the instruction in program being executed ---Register file – state element that consists a set of registers that can be read and written by supplying register number to be accessed ---Sign extend – replicating high order sign bit, used to increase size of number ---Branch target address – becomes new program counter if taken, given by sum of offset and address of instruction ---Branch taken – branch condition satisfied, pc becomes branch target. All unconditional jumps are this type ---Branch not taken – branch condition false, pc becomes address of the instruction sequentially following the branch ---Branch – type of branch where instruction immediately following the branch is always executed. ---Don’t care term – element of a logical function where output does not depend on inputs ---Opcode – field that denotes the operation and format of an instruction ---Single cycle implementation – instruction executed in one clock cycle, to slow to be practical ---Pipelining – multiple instructions overlapped in execution ---Structural hazard – planned instruction cannot execute in proper clock cycle because hardware does not support the combination of instructions that are set to execute (using a combo dryer and washer) ---Data hazard – instruction cannot execute because data is not available (can still cause bubble / forward stall) ---Forwarding – method for resolving data hazards by retrieving the missing element ---Load use data hazard – data used by load has not become available (data hazard not fixed by forwarding) ---Pipeline stall bubble – stall initiated in order to resolve a hazard ---Control hazard –need to make decision from event still in execution (football team laundry example, predict correct formula and run two loads together assuming correct formula, fix later if not) ---Branch prediction – resolves branch / control hazard by assuming a given outcome ---Prediction - ---Parallelism – fundamentally invisible to the programmer ---Latency – number of stages between two instructions ---Nop – instruction that does no operation to change state ---Flush – discard instructions in a pipeline due to unexpected events ---Dynamic branch prediction – assumption at runtime with runtime info ---Branch prediction buffer – small portion of memory that indicates if recent braches were taken or not ---Branch delay slot – slot directly after delayed branch ---Branch target buffer – caches the destination PC or destination instruction of a branch ---Correlating predictor – local and global information on branches ---Tournament branch predictor – multiple predictions available and selector to choose ---Exception – unscheduled event that disrupts program ---Interrupt – exception from outside processor ---Vectored interrupt – an interrupt for which the address to which control is transferred to display cause of interrupt ---Imprecise interrupt – not associated with the exact instruction that caused exception ---Instruction level parallelism – parallelism among instructions ---Multiple issue – a scheme where multiple instructions are launched in one clock cycle ---Static multiple issue – many decisions made by compiler before execution ---Dynamic multiple issue – decisions during execution by compiler ---Issue slots – positions from which instructions could issue in a clock cycle ---Speculation – compiler or processor guesses outcome ---Issue packet – set on instructions grouped in one clock cycle. Static compiler or dynamic processor decision ---Very long instruction word – style of ISA that launches wider commands. ---Use latency – number of cycles between load and instruction on result of load ---Loop unrolling – more performance from loop, multiple copies of loop body created ---Register renaming – remove antidependencies, either hardware or software ---Antidependence – caused by reuse of a name, not true dependence between two source items ---Superscalar – enables processor to execute more than one instruction per clock cycle ---Dynamic pipelining scheduling – hardware reorders instructions to avoid stalling ---Commit unit – decides when to release programmer visible registers and memory when it is safe ---Reservation station – buffer in functional unit that holds operands and operators ---Reorder buffer – holds results of dynamically scheduled processor until is it safe to store ---Out of order execution – blocked instruction that does not cause following instructions to wait ---In-order commit- results of pipelined execution to a state the programmer can see in the same order the instructions are fetched**Icons and chapter concepts**

Core instruction set:

Memory reference instructions load word and store word

Arithmetic logical instructions add,sub, AND, OR and slt

Branch equal and jump

Key design principles:

Simplicity favors regularity

Implementation overview:

1. Send PC to location that contains the code and fetch instruction from there
2. Read registers (1 or 2 of them)
3. Now instructions split paths, but they all use ALU (arithmetic logical unit) except jumps
4. Memory reference – access memory to read data for load or write from ALU or mem back to register
5. Arithmetic logical or load – write data from ALU or mem to register
6. Branch – may need to change address based on comparison, otherwise increment PC by 4 to go to next instruction

Five classic components of computer:

Control, datapath, memory, input, output

Edge triggered clocking:

[State element] -> (combinational logic) -> [back into initial state element]

Can read and write in same cycle, but feedback cannot occur within same cycle

Store and access instructions:

Instruction memory (read location of instruction) and program counter (to hold this memory item)

Adder (ALU designed to only add) is required after these two to find next instruction

R type:

Register file and ALU

Load and store:

Register file, ALU, data memory unit and sign extension unit

Single cycle data path needs separate memory and data because processor operates in one cycle so two memories required.

Instruction formats:

R

31:26 is op, 25:21 is rs base regiser, 20:16 is rt second base regiser, 15:11 is rd destination register. 10:6 is shamt, 5:0 is funct

Load

31:26 is op, 25:21 is rs, 20:16 is rt destination register, 15: 0 is address

Branch

31:26 is op, 25:21 is rs, 20:16 is rt, 15: 0 is address

Control Signals 264 \*\*

IF: Instruction Fetch, ID: Instruction Decode and Reg file read, EX: execution or address calc, MEM: Data memory access, WB: Write Back

Draw fig 4.33

Only challenge to forwarding is to op in EX stage that a earlier instruction intends to write in it’s WB stage

ILP – divide washer into 3

**Big Picture**

Pipelining increases number of commands running in parallel and the throughput of commands. No effect on latency.

Compiler relies on hardware to resolve hazards, but compiler still must know what’s going on to operate correctly and with best performance

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Sig | RegDest | RegWrite | ALUSrc | PCSrc | MemRead | MemWrite | MemtoReg | | 0 (deas) | Reg dest becomes rt (20:16) | None | 2nd alu op comes from read data 2 | PC replaced by adder PC+4 | None | None | Value to reg write data input from ALU | | 1 (assert) | Reg dest becomes rd (15:11) | Reg on write data input is written with write data input | 2nd ALU op is sign ext | PC replaced by output of adder (branch targ) | Data memory from address input put on read data output | Data mem from address input replaced by value on Write data input | Value to reg write data input from the data memory | |

After successful completion of this course, the student should:

2. Have an ability to represent numbers in and convert between decimal, binary, and hexadecimal and perform calculations using 2’s complement arithmetic.

Decimal to binary – divide by two, write remainder off to side, continue until quotient is 0. Binary number is the remainders in reverse order. Add any extra bits to format correctly. Invert all bits for 1’s compliment. Add an extra 1 to that number for 2’s compliment.

Decimal to hex – divide by 16, write remainder off to side, continue until quotient is 0. Hex number is the remainders in reverse order, converted to hex equivalent. Invert each hex for 1’s. Add one for 2’s

Binary to decimal – line add bits up in left column, make a sum column next to that, make a mult column next, make a product column. Make all mult column 2’s. Move first bit into sum column, multiply sum and mult column and store in product. Now on next line the sum is the current bit value plus the previous product. Multiple new sum and mult column for current product. Repeat until last sum column. Do not multiply on very last row.

Hex to decimal – if first digit in 2’s compliment is 9-F then number is negative. Take 1st nibble, convert to decimal, multiply by 16. Convert 2nd nibble. Add to sum. Sum \* 16. Convert 3rd nibble. Add to sum. Sum \* 16

Final 2 Problems, Chapters 2-3

3. Understand the basic model of a computer including the datapath, control, memory, and I/O components.

5 classic components - input, output, memory, datapath, control, (datapath + control) = processor

Final, Chapters 1, 4, & 5

5. Understand the role of compilers, assemblers, and linkers and how programs are translated into machine language and executed.

Compiler - translates high level code into assembly language

Assembler - converters assembly language into binary machine code

Linker –

Row for fig A-8

Final, Chapters 1 and Appendix A

6. Be able to demonstrate comprehension of a pipelined architectures including datapaths and hazards.

Final, Chapter 4

7. Be able to demonstrate comprehension of computer performance measures and their estimation.

Response time, wallclock time and elapsed time - time to complete task. Decreasing response time almost always increases throughput.

Throughput- number of tasks completed for unit time. Sometimes increasing throughput can reduce response time if there are items waiting in que.

Latency – changing latency has no effect on either response time or throughput

CPU Execution – time cpu spends computing

User cpu time – time cpu spent in program

System cpu time – time cpu spent in os on behalf of program

CPI – average number of clock cycles per instruction

Time = Seconds / Program = (Instructions / Program) \* (Clock cycles / instruction) \* (Seconds / Clock Cycle)

Final, Chapter 1

8. Understand the memory hierarchy including caches and virtual Memory

Two rows for page 375 pic

Final, Chapter 5

Temporal Locality – anything referenced once will be needed again soon

Spatial locality – anything around something referenced will be needed soon

Memory hierarchy – multiple levels of memory, as distance from processor increases so does size and access time

Block – minimum unit of information in a cache

Hit rate – fraction of accesses found in level needed

Miss rate – accesses not found

Hit time – time required to access

Miss penalty – time required to fetch block from the lower level to the level needed, time to access block, transmit between levels, insert into level that missed, pass to requestor.

Track – one of thousands of circles making up a disk

Sector – one of segments that make up a track, smallest amount of info read or written to disk

Seek – processor of moving read write head to proper track

Rotational latency – time required to move head

Direct mapped cache – each memory location mapped to one location in cache

Tag- field in table for memory hierarchy that contains address info need to determine if block in hierarchy corresponds to the word

Valid bit- field in memory table that designates if the contents are valid

Cache miss – request for data that is not available

Write through – writes always update both cache and next level lower. Consistent data

Write buffer – que that holds data to be written

Write back – updates values only in cache, writes modified block to lower level when the block is replaced

Split cache – one level with two caches one for data and one for instructions

Fully associative – block can be placed anywhere in cache

Set associative – fixed locations where block can be placed in cache

Least recently used- replace block that has been used the longest

Global miss rate – misses in all levels

Local miss rate – misses on one level

Error detector code- identifies error but no location or solution

Virtual memory – main memory as “cache” for seconday storage

Physical address – address in main memory

Protection- mechanisms to ensure multiple processes cannot interfere with eachother. Isolate OS from user process

Page fault – accessed page not available

Virtual address – corresponds to virtual location that is mapped to physical when requested by address mapping

Address translation – process of mapping virtual to physical address

Segmentation – variable size mapping , segment number which correlates to a physical address and segment offset.

Page table – table holding virtual translations indexed by virtual page number

Swap space – space on disk reserved for VM space of a process

Referenced bit – set when page is accessed to implement LRU

Translation lookaside buffer – cache keeps track of recently used addressed to avoid going to page table

Virtually addressed cache – accesed with virtual address rather than physical

Aliasing – two addresses point to same object

Physically addressed cache – addressed by actuall memory locations

System call – passes control from user to supervisor code space that starts the required exception for the call

Context switch – changing processor state to allow new process to use, saveing current state as well.

Exceptional enable – interupt enable, signal that determines response to exception. Handles saving before acting

Restartable instruction – can resume after expection is resolved

Hanlder – name of software routine to deal with exception or interrupt

Unmapped – portion of page space that cannot have page faults

Three c’s – complosury misses, capacity misses, conflict misses

Compulsory – block never in cache before

Capacity – even with full associativity, cache can’t hold all blocks needed

Conflict – set associative or direct, multiple blocks compete for same set