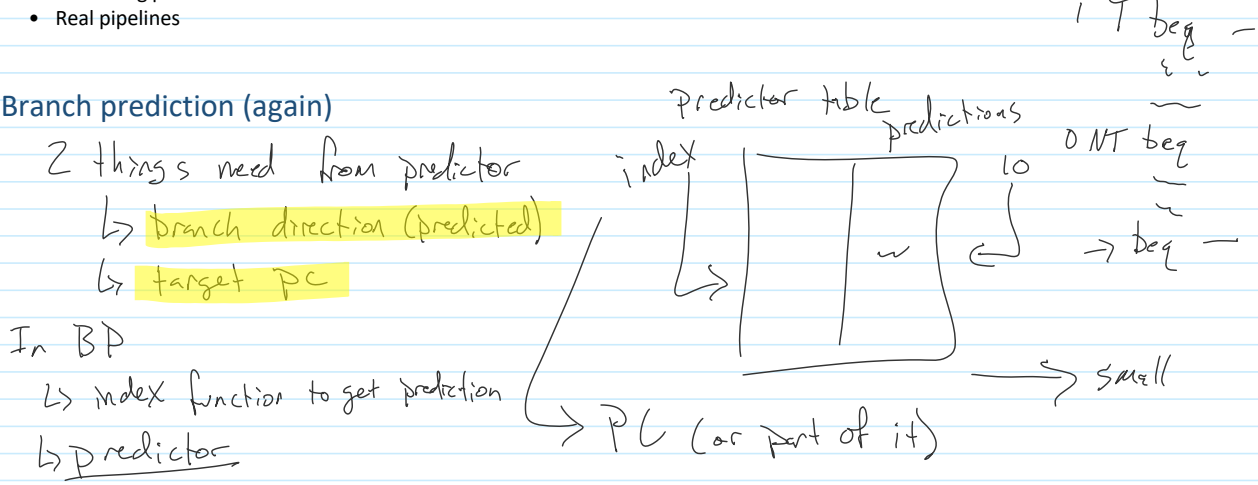
Lecture 10: Correctness and ILP

Wednesday, February 7, 2018 5:51 PM

Outline

- Finish branch prediction
- Exceptions
- Increasing performance
- Real pipelines

Branch prediction (again)



2 things need from predictor
- branch direction (predicted)
- target pc

In BP
- index function to get prediction
- predictor
  - Saturating counter
  - 1 bit
  - general state machine
  - neural network (perceptron)

or global branch history

→ must be very fast
- limits complexity

★ last necessary bit
update prediction

Exceptions

- special case
  - → something "unexpected" or incorrect happens
  - → interrupts
  - → undefined inst.
  - → divide by 0
  - → out of bounds memory access / seg fault

What's needed to support exceptions?

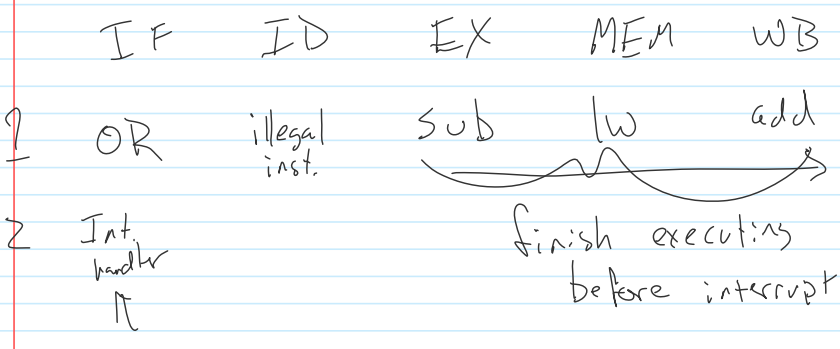
redirect control to a function
interrupt/exception handler
- part of the OS

Save state
- PC → allows us to return

What caused the exception

Jump to the OS

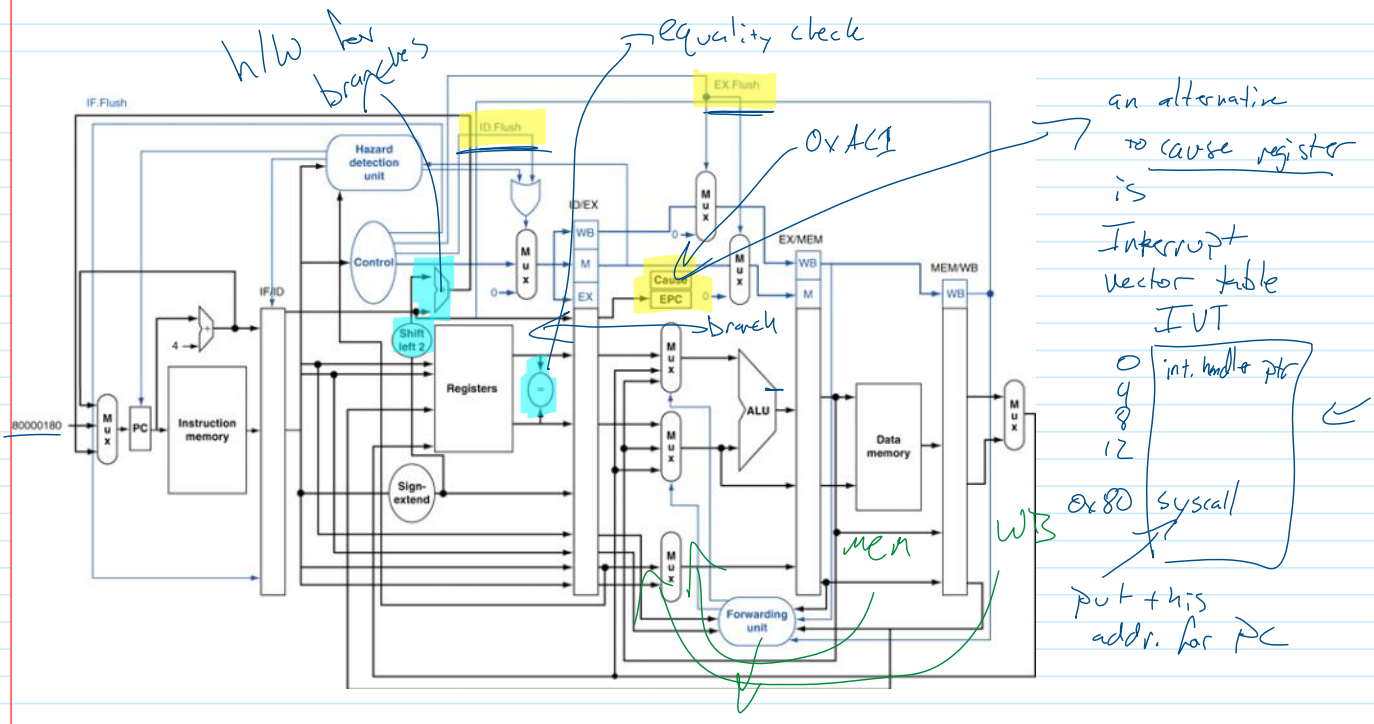
In MIPS two registers
- EPC → excepting PC
- cause



Precise exceptions: appear as if executed on a single cycle machine

11

Precise exceptions: appear as if executed on a single cycle machine



Increasing performance

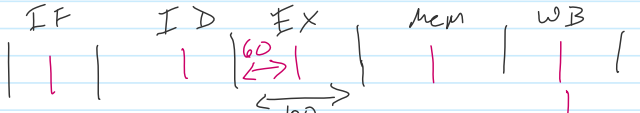
20-30 billion (10^9) inst. pr second (one core) →

Multiple inst. per cycle → "instruction-level parallelism"
↳ compilers
↳ H/W

Multiple processors → skip this

add more stages → shorter stage → higher freq. simple pipelining → CPI = 1

incr freq → incr. perf.



- need to find "intermediate" results → extra buffers → take time → limit to freq incr.
- split longest first
  ↳ affects cycle time most
  → goal is for all stages equal time → balanced pipeline

biggest complication

hazard detection
forwarding very complex
memory problem → can't pipeline
precise exceptions

★ flushing is huge overhead on wrong branch prediction

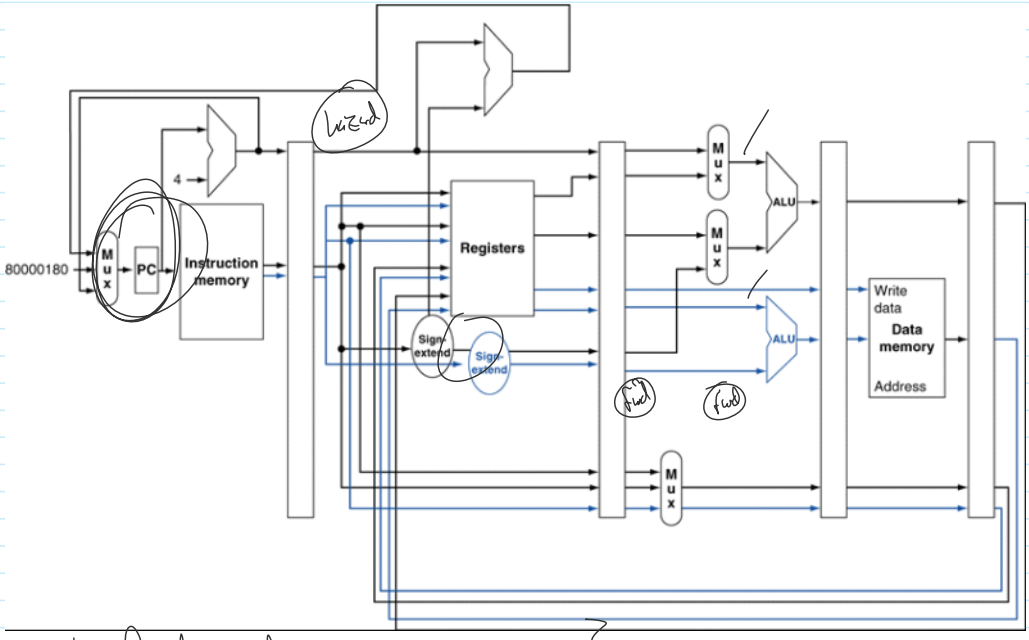
how? multiple inst. per cycle?

fetch 2 inst.
decode " " → double the pipeline
ex " "

Hazards between 2 simultaneous inst.
↳ double → Area overhead ↑ 4x

add $1, $2, $3
lw $7, 100($6) } independent

Inst level parallelism
→ parallelism in one thread
→ in one inst stream



how to find independent instructions?

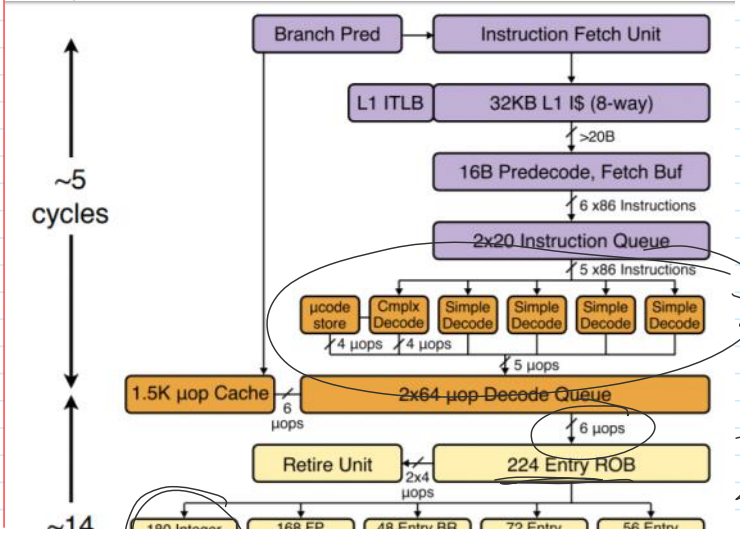
We can use hardware
or → compiler
↳ reorder & find indep. inst. → pack them into blocks

VLIW
↳ very long inst word

for i 1..n
beq
unroll the loop
for i 1..[n/2]
fetch
beq

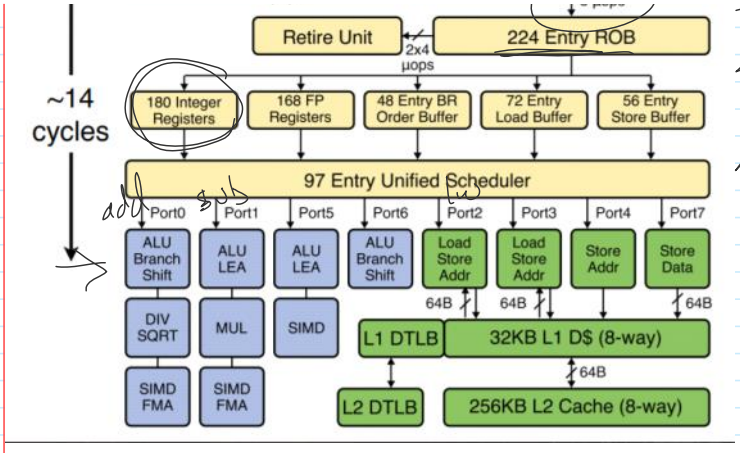
Real pipelines

Intel skylake

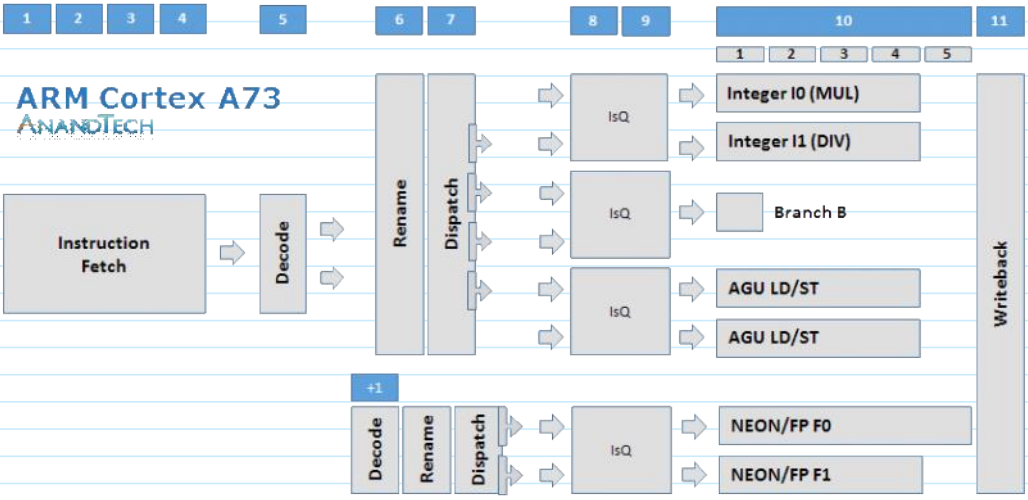


decode 20 cycle
breaks down inst. into µ-ops

Out-of-order


~14 cycles
Retire Unit
2x4 µops
224 Entry ROB
180 Integer Registers
168 FP Registers
48 Entry BR Order Buffer
72 Entry Load Buffer
56 Entry Store Buffer
97 Entry Unified Scheduler
Port0
Port1
Port5
Port6
Port2
Port3
Port4
Port7
ALU Branch Shift
ALU LEA
ALU LEA
ALU Branch Shift
Load Store Addr
Load Store Addr
Store Addr
Store Data
64B
64B
64B
DIV SQRT
MUL
SIMD
SIMD FMA
SIMD FMA
L1 DTLB
32KB L1 D$ (8-way)
64B
L2 DTLB
256KB L2 Cache (8-way)
Out-of-order
WB
x86 → 16 reg.
exec/mm

Arm A73


1
2
3
4
5
6
7
8
9
10
11
ARM Cortex A73
AnandTech
Instruction Fetch
Decode
Rename
Dispatch
IsQ
IsQ
IsQ
Integer I0 (MUL)
Integer I1 (DIV)
Branch B
AGU LD/ST
AGU LD/ST
Writeback
+1
Decode
Rename
Dispatch
IsQ
NEON/FP F0
NEON/FP F1