Comparative Architectures

Source

- IB Architecture / Computer Design
- IB Compiler Construction
- II Advanced Computer Architecture / Comparative Architectures
- Computer architecture: a quantitative approach
    - Hennessy, J.L. & Patterson, D.A (2011)

Analogue and digital

- What are the advantages and disadvantages of analogue computers over their digital counterparts?

	analogue (oscilloscope)	**digital computer**
feature	continuous values / physical data	discrete values / binary system
speed	slow	fast
memory capacity	low or limited	large
reliable / accurate	no (checksum)	yes
usage, arch	complicated	easy
result	voltage signals	computer screen
energy	current -- power-hungry	lower power
reprogram	wirable	reconfigurable
communication	radio signal (speed)	bus, wire (1/10 speed of light)

Relationship: Analog = Quantize [0,255] saturated by Max $\implies$ Digital

Digital computer system comes from analog and the conversion has a cost.

Digital one has repeatable complex components, Inductor.

Modern Compiler

Key takeaway from translator (interpreter) shown in Compiler Lecture,

- Divide from single into two stages
    - Compile (inspect)
    - Interpret (compute)

- Divide from single into two stacks (memories)
    - instruction stack / IM

```
PUSH, POP, MK_PAIR
```

    - data value stack / DM
- Separation of the two memories (Instruction and Data)
    - allows for simultaneous access
        - an instruction can be read while a data memory is read or written in the same cycle.
        - Motivation for pipeline and multi-issue *superscalar* (Instruction level parallelism)
        - more difficult with a unified cache/memory.
    - instruction memory is read-only and has less circuitry
        - has no dirty bits, no write back, etc
        - the IM and DM can have different associativity
    - Downside: von Neumann bottleneck
        - common bus (address, data, control)

Aside: Turing tax (universal computing machine) vs special purpose processor.

Flynn's Taxonomy

Based on parallelism on instruction and data streams, the first four kinds listed below

SISD

- A simple processor

MISD

- Used for redundancy
    - Fligh control system, error-detection

SIMD

- Vector processing
    - Vector registers each hold several data items
        - hardware: Regs = Reg $\times n$
    - Vector operations (add, multiple)
        - hardware: ALU $\times n$
- Energy-efficient, data level parallelism

MIMD

- Multicore, standard general purpose CPUs

Extra: SIMT

- each thread has *separate state* (registers and memory)
    - e.g. stack pointer (sp)
- data level parallelism

Processor					**Note**
instruction	---S---	Fetch	Decode	---S---	*Shared*
memory	---S---	Shared	Memory	---S---	*Shared*
processing	ALU0	... ...	... ...	ALU31	*single 32-value vector operation*
thread states	Regs 0	Regs 1	Regs 2	Regs 3	*each vector register contain 32 floats*
	... ... Regs 12	... ... Regs 13	... ... Regs 14	... ... Regs 15	*hide latency when stall*
thread context	T0	T1	T2	T3	only**one** run *save context switch*

Architectures comparison

Source: Classifying Instruction Set Architectures (Textbook **A.2**)

Architecture	**Accumulator**	**Stack**	**Register File**
operands: from memory and	acc + 3 = 4	top of the stack	rs1, rs2 (disjoint), rd orthogonal needs less
instruction density	shortest less mem space	concise (short instr)	longer
von Neumann bottleneck (Mem bus)	worse for mem (RTT) mem bus 2x CPU ⟺ 2x frequency	store imm in stack (near) If stack is full, memory	store in cache (nearer) fast mem access ⟺ higher frequency
caching	hard to predict	predictable	in the middle
power consumption	less few memory accesses	less for control few memory accesses	most multi-issue
multi-issue	0	0	Yes
performance	Calculator ENIAC	razer printer, compiler(JVM) Hard for queue, list, swap	modern CPU IC best

	superscalar	**compiler VLIW**	**SIMD**	**multi-core**	**DSA**
parallelism	static or dynamic ILP	static ILP	DLP	TLP	custom
features	instruction fetch, dynamic prefetch, memory addr. alias, physical regs. rename	scheduling; sw. speculate; var. /fixed bundle	N ops., independent, same FU, disjoint regs., known mem access,	fine/coarse-grained vs. SMT	specialized
instr. count (IF/DE)	↑ out-of-order	one VLI	↓	var.	custom
branch handling	dynamic branch pred.	limited	poor (predicted)	per-core	custom
limitations	fabrication, and below	tailor to a pipeline	data-level tasks	Amdahl's Law	inflexible
hardware cost/area	↑	↓	vector regs.	pipeline regs.	custom
interconnect	↑ (in single core)	↓	wide data bus, lane	mesh/cache coherence	scratchpad
energy cost	↑	↓	↓	var.	↓ ✰
binary compatibility	✓	×	▢ (✓ VLA)	✓	×
use cases	CPU, general-purpose	embedded, GPU	ML, graphics	CPU, server, SoC	TPU, DSP

Addressing and cache

	virtual addressing	**physical addressing**
index / hit time	fast, check within offset permission check TLB later	slow, wait translation south bridge hw (address space)
address after context switch	same virtual for different physical addresses	different physical addresses
prefetchable	Yes	no (update rather than cache)
aliasing (different virtual)	yes (coherence problems)	No
others	homonyms problem (different physical if not flush)	network package (last bit) not write mergeable (two core write)

Depending on the index and tag addressing mode:

- VIPT, VIVT, PIPT, PIVT cache