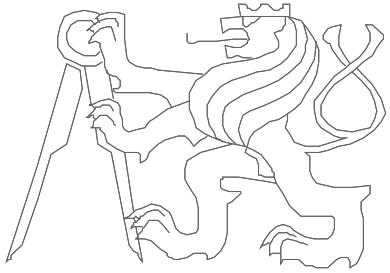
Computer Architectures

Microprocessor Evolution - from 4-bit Ones to Superscalar RISC

Pavel Píša, Michal Štepanovský



Czech Technical University in Prague, Faculty of Electrical Engineering

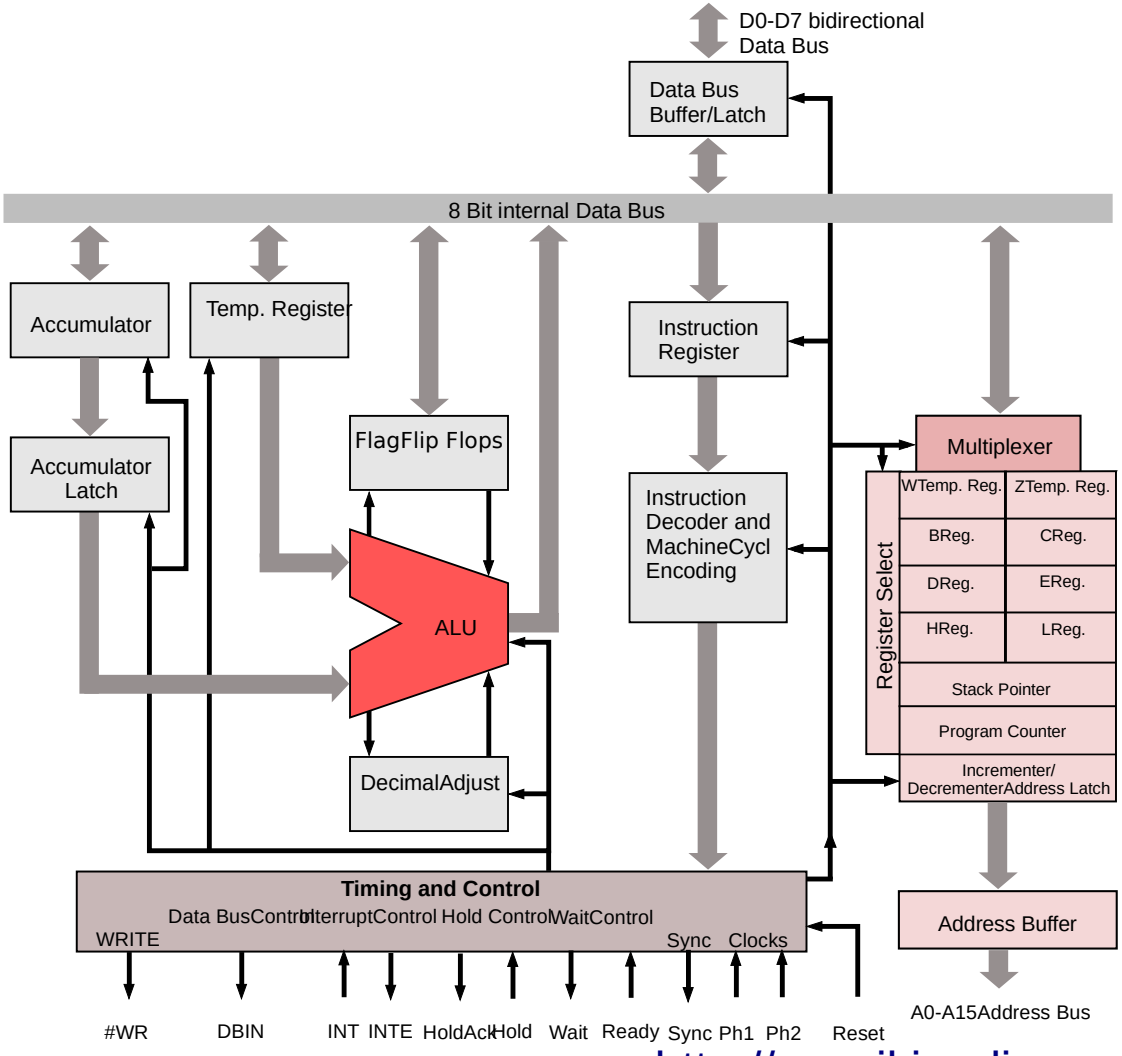
Early Technology and Complexity Comparison

CPU	Company	Year	Transis.	Technology	Reg/Bus	Data/prog+IO	Cache I/D+L2	Float	Frequency	MIPS	Price
4004	Intel	1971	2300	10um - 3x4mm	4bit	1kB/4kB			750kHz	0.06	200
8008	Intel	1972	3500	10um	8bit	16kB				0.06	
8080	Intel	1974	6000	6um	8bit	64kB+256			2MHz	0.64	150
MC6501	NMOS T.	1975									20
8085	Intel	1976	6500	3um	8bit	64kB+256			5MHz	0.37	
Z-80	Zilog	1976			8bit	64kB+256			2.5MHz		
MC6502	NMOS T.	1976									25
8086	Intel	1978	29000	3um	16/16bit	1MB+64kB			4.77MHz	0.33	360
8088	Intel	1979		3um	16/8bit	1MB+64kB			4.77MHz	0.33	
MC68000	Motorola	1979	68000		16-32/16bit	16MB					
80286	Intel	1982	134000	1.5um	16/16bit	16MB/1GBvirt	256B/0B		6MHz	0.9	380
MC68020	Motorola	1984	190000		32/32bit	16MB	Ano		16MHz		
80386DX	Intel	1985	275000	1.5um	32/32bit	4GB/64TBvirt			16MHz		299
MC68030	Motorola	1987	273000			4GB+MMU	256B/256B				
80486	Intel	1989	1.2mil	1um	32/32bit	4GB/64TBvirt	8kB	Ano	25MHz	20	900
MC68040	Motorola	1989	1.2mil			4GB+MMU	4kB/4kB	Ano			
PowerPC 601	Mot+IBM	1992	2.8mil		32/64bit	'2^{56}	32kB	Ano	66MHz		
PA-RISC	HP	1992							50MHz		
Pentium	Intel	1993	3.1mil	0.8um - BiCMOS	32/64bit	4GB+MMU		Ano	66MHz	112	
Alpha	DEC	1994	9.3mil		64bit	4GB/64TBvir	8/8+96kB		300MHz	1000	
MC68060	Motorola	1994	2.5mil			4GB+MMU	8kB/8kB	Ano	50MHz	100	308
Pentium Pro	Intel	1995	5.5mil					Ano	200/60MHz	440	1682
Pentium II	Intel	1998	7.5mil		32/64bit			Ano+MMX	400/100MHz	832	
PowerPC G4M PC7400	Motorola	1999		0.15um – cooper 6LM CMOS	64/128bit	4GB/2^{52}	32kB/32kB +2MB	Ano+AV	450MHz	825	

Accumulator Based Architectures

- register+accumulator → accumulator
  - 4bit Intel 4004 (1971)
  - 8bit Intel8080 (1974) – registers pairs used to address data in 64kB address space
  - basic arithmetic-logic operations only – addition, subtraction, rotation for 8-bit accumulator
  - subroutines by CALL and RET instructions with 16-bit PC save on stack
  - a few 16-bit operations – increment/decrement of registers pairs, addition to HL and save to stack
  - microcode controlled/microprogrammed instructions execution – 2 to 11 clock cycles per instruction at 2 MHz clock signal

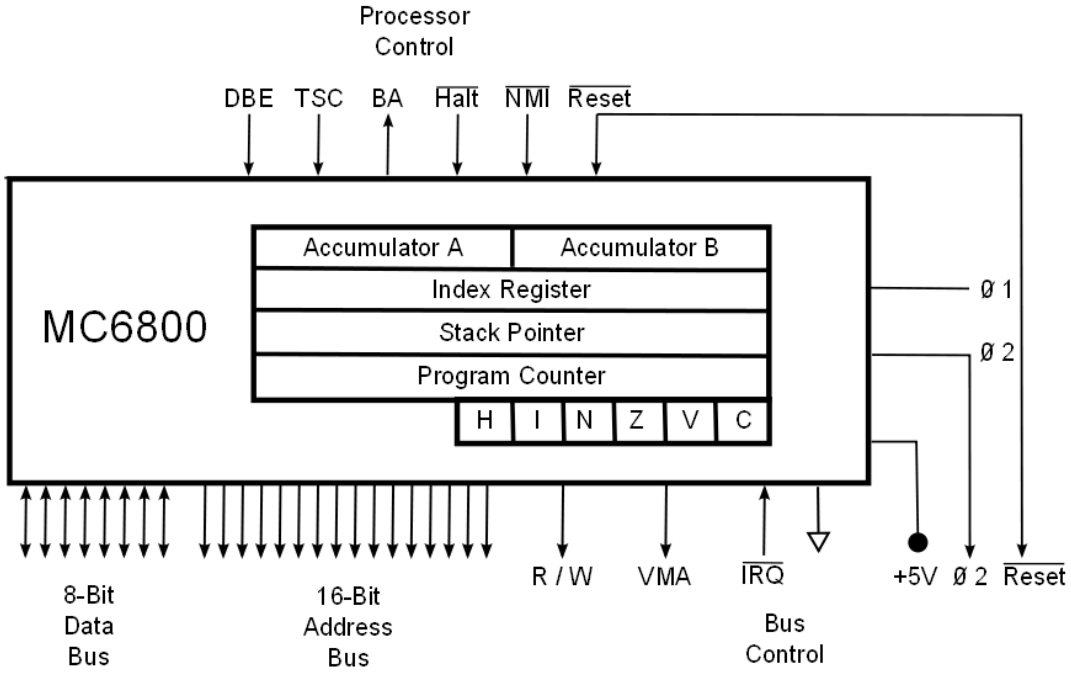
Intel 8080



http://en.wikipedia.org/wiki/Intel_8080

Fast memory ⇒ reduce register count and add address modes

- Motorola 6800, NMOS T. 6502 (1975) - accumulator, index, SP a PC only – use zero page as fast data, CU hardwired
- Texas TMS990 – workspace pointer only, even PC, SP, other registers in main memory, similar to transputers

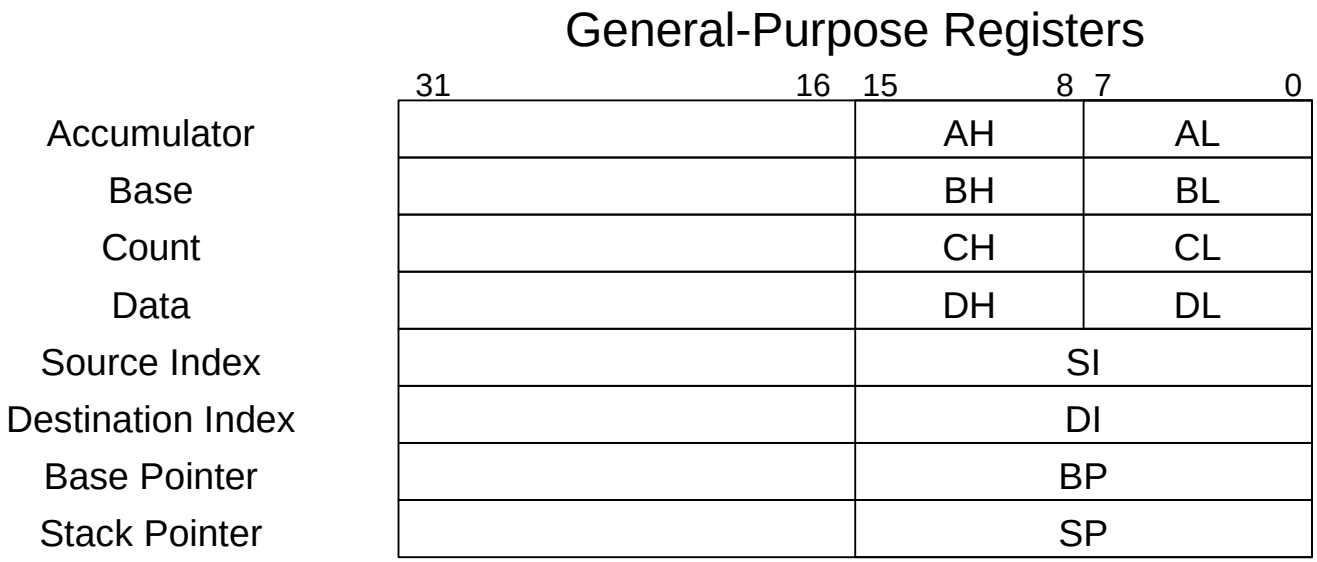


Memory is bottleneck now ⇒ complex instruction set modeled according to C language constructs, CISC

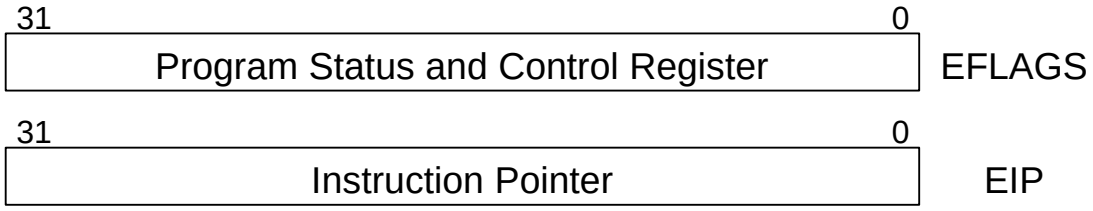
- Intel 8086 (16-bit upgrade to 8080)
  - 8× 8-bit register form 4 pairs (16-bit registers), additional 4 16-bit registers – SP, BP (C functions frame), SI (source-index), DI (destination index), 1 MB address space by segments, register+=register, memory+=register, register+=memory
- Motorola 68000 (1979) – 16/32bit
  - two operand instructions
  - register+=register, memory+=register, register+=memory, even one instruction memory=memory
  - based on microcode to process so rich instruction set
- Z-8000 16bit, Z-80000 32bit (1986) CISC
  - 6 phases pipelined execution, without microcode, 18000 transistors only

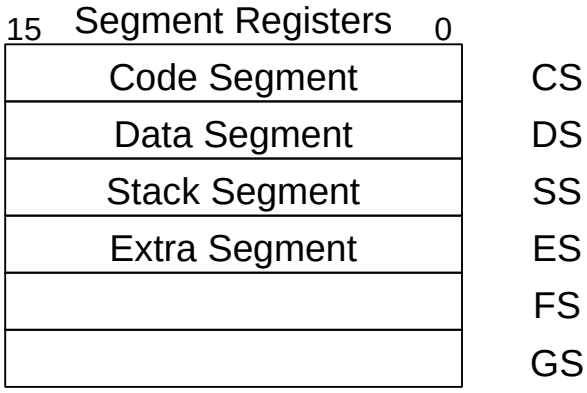
Intel 8086 and 32-bit i386

General-Purpose Registers



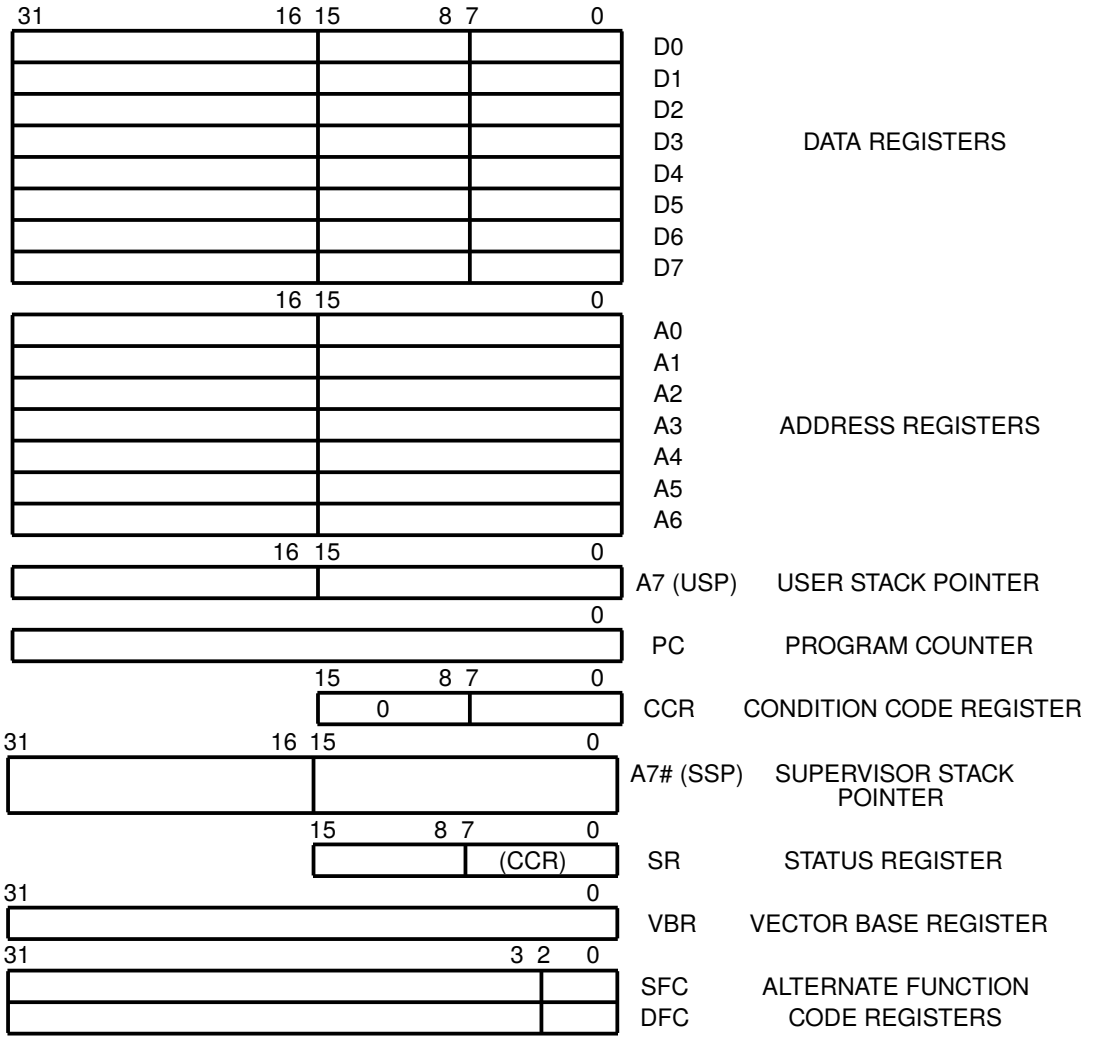
	16-bit	32-bit
Accumulator	AX	EAX
Base	BX	EBX
Count	CX	ECX
Data	DX	EDX
Source Index		ESI
Destination Index		EDI
Base Pointer		EBP
Stack Pointer		ESP





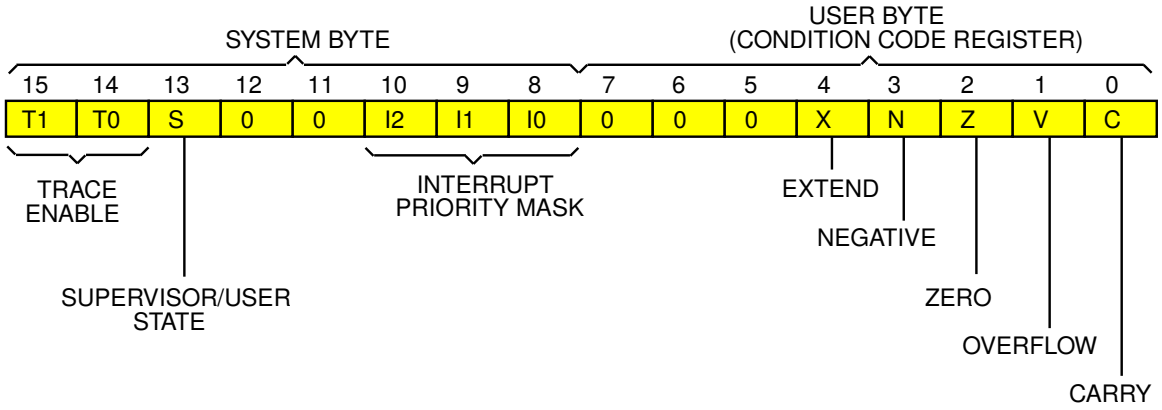
Basic Integer Registers of M68xxx/CPU32/ColdFire

User mode



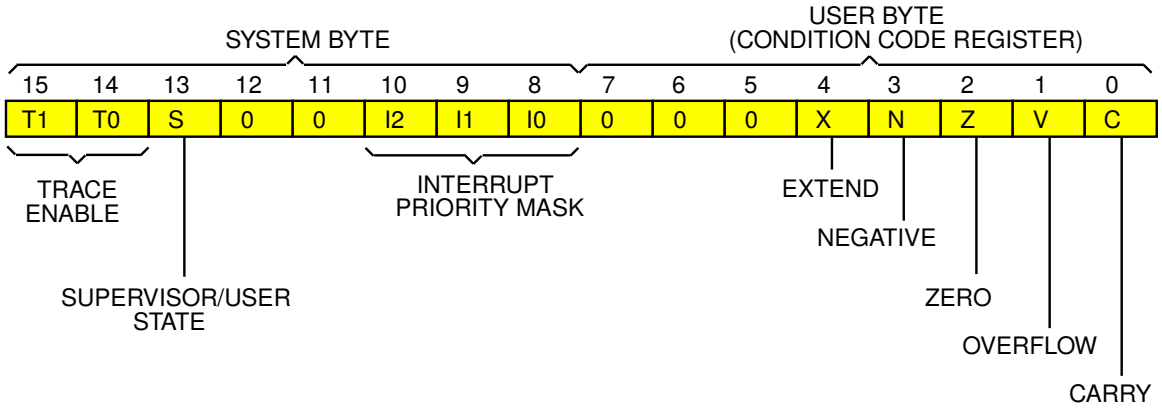
System mode

Status Register – Conditional Code Part



- N – negative ... = 1 when the most significant bit of the result is set; otherwise cleared. (the result is negative for two's complement representation)
- Z – zero ... = 1 when result is zero – all bits are zero
- V – overflow .. = 1 when an arithmetic overflow occurs implying that the result cannot be represented in the operand size (signed case for add, sub, …)
- C – carry ... = 1 when when a carry out of the most significant bit occurs (add) or a borrow occurs (sub)
- X - extend (extended carry) .. Set to the value of the C-bit for arithmetic operations; otherwise not affected or set to a specified result

Status Register – System Byte



- T1, T0 – trace … if some of these bits is set then exception is generated after every instruction execution or when program flow changes (jump, call, return)
- S – supervisor … if set to 1 then CPU runs in the supervisor state/mode and SP maps to SSP. Else CPU runs in user mode, SP maps to USP and changes to the system byte are not possible and user mode privileges rules/restrictions are applied to memory access (controlled by MMU).
- I2, I1, I0 - interrupt mask … up to this interrupt priority level are requests blocked/masked – i.e. they need to wait. The level 7 is exception because it is non-maskable, i.e. exception acceptance cannot be delayed.

Addressing Modes – Basic 68000 Modes

- **Up to 14 addressing modes for operand selection**
- **Rn** operand represents value of data Dn or address An register
- **(An)** memory content at the address specified by **An**
- **(An)+** memory content at **An** with following **An** increment by value equivalent to the operand length (post-increment)
- **-(An)** the **An** register is decremented by operand size first and then specifies memory located operand (pre-decrement)
- **(d16,An)** memory at **An** + 16-bit sign extended offset
- **(d8,An,Xn)** memory at **An** + 8-bit sign extended offset + index register (another **Am** or D**m**) which can be eventually limited to lower 16 bits, index can be multiplied by 1, 2, 4 or 8 for CPU32 and 68020+ processors
- **(xxx).W** 16-bit absolute address – upper and lower 32kB
- **(xxx).L** 32-bit absolute address

Data throughput and instruction fetching slow still ⇒ cache memory

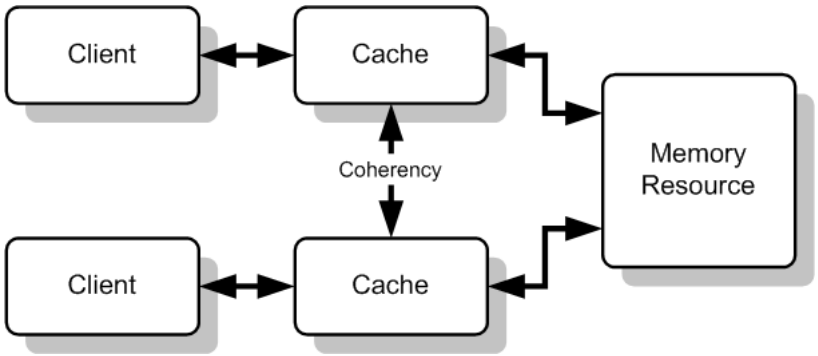
- The problem has been solved quite well
- Common cache or Harvard arrangement I & D
- More levels (speed limited for bigger size – decoder, capacitance of common signals)
- But requires to solve data coherence when DMA access or SMP is used
  - synchronization instructions for peripherals access and synchronization `eieio` (PowerPC), `mcr p15` (ARM), …
  - hardware support required for caches and SMP
    - protocol MSI , MESI (Pentium), MOSI
    - MOESI AMD64 (Modified, Owned, Exclusive, Shared, and Invalid)

Data Coherence and Multiple Cached Access

MOESI protocol

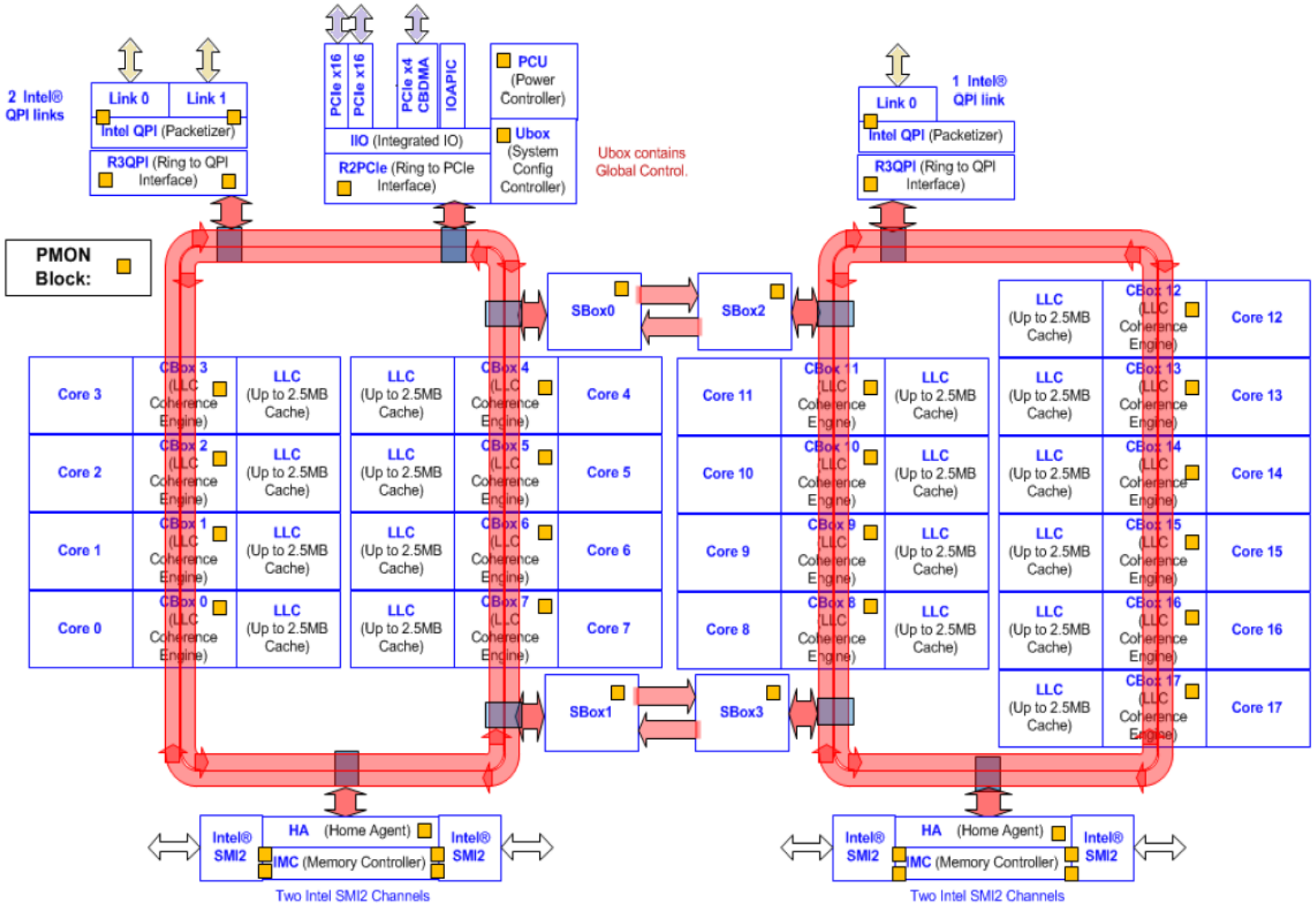
- Modified – cache line contains actual and modified data, none of other CPUs works with data, old/previous data are hold in main memory
- Owned – line holds actual data, line can be shared with other CPUs CPU but only in S state, main memory is not required to be up to date
- Exclusive – only this CPU and main memory contains cahe line data
- Shared – cache line is shared with other CPUs, one of them can be in O state, then data can differ to content in main memory
- Invalid – cache line does not hold any valid data

	M	O	E	S	I
M	N	N	N	N	Y
O	N	N	N	Y	Y
E	N	N	N	N	Y
S	N	Y	N	Y	Y
I	Y	Y	Y	Y	Y



http://en.wikipedia.org/wiki/MOESI_protocol

Supercomputers NUMA on Chip – Broadwell-EP (Intel Xeon)

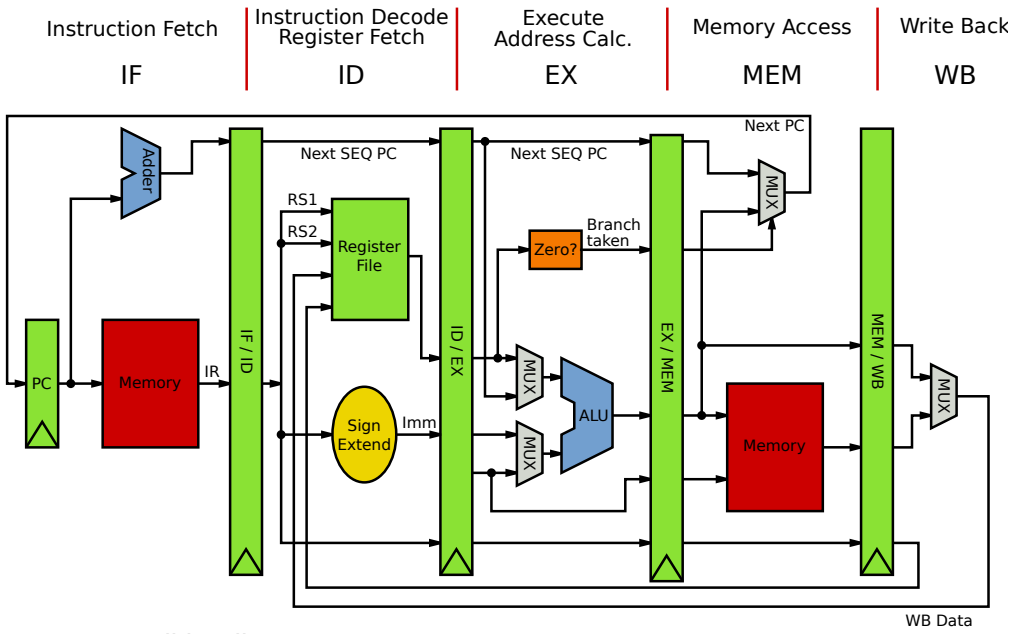


Yet faster instructions execution ⇒ RISC architectures

- Reduce data flow dependency between instructions, three operand instructions, speculative instructions execution, more registers to reduce memory accesses, register renaming, eliminate interdependencies on conditional code/flag register (MIPS RISC-V eliminate CC altogether, DEC Alpha, multiple flag registers PowerPC, flags update suppress ARM)
- load-store architecture, computation only register+=register and or register=register+register and separate load-store instructions.
- Fixed instruction encoding ⇒ programs are usually longer but much faster instructions decoding, optimized for pipelined execution

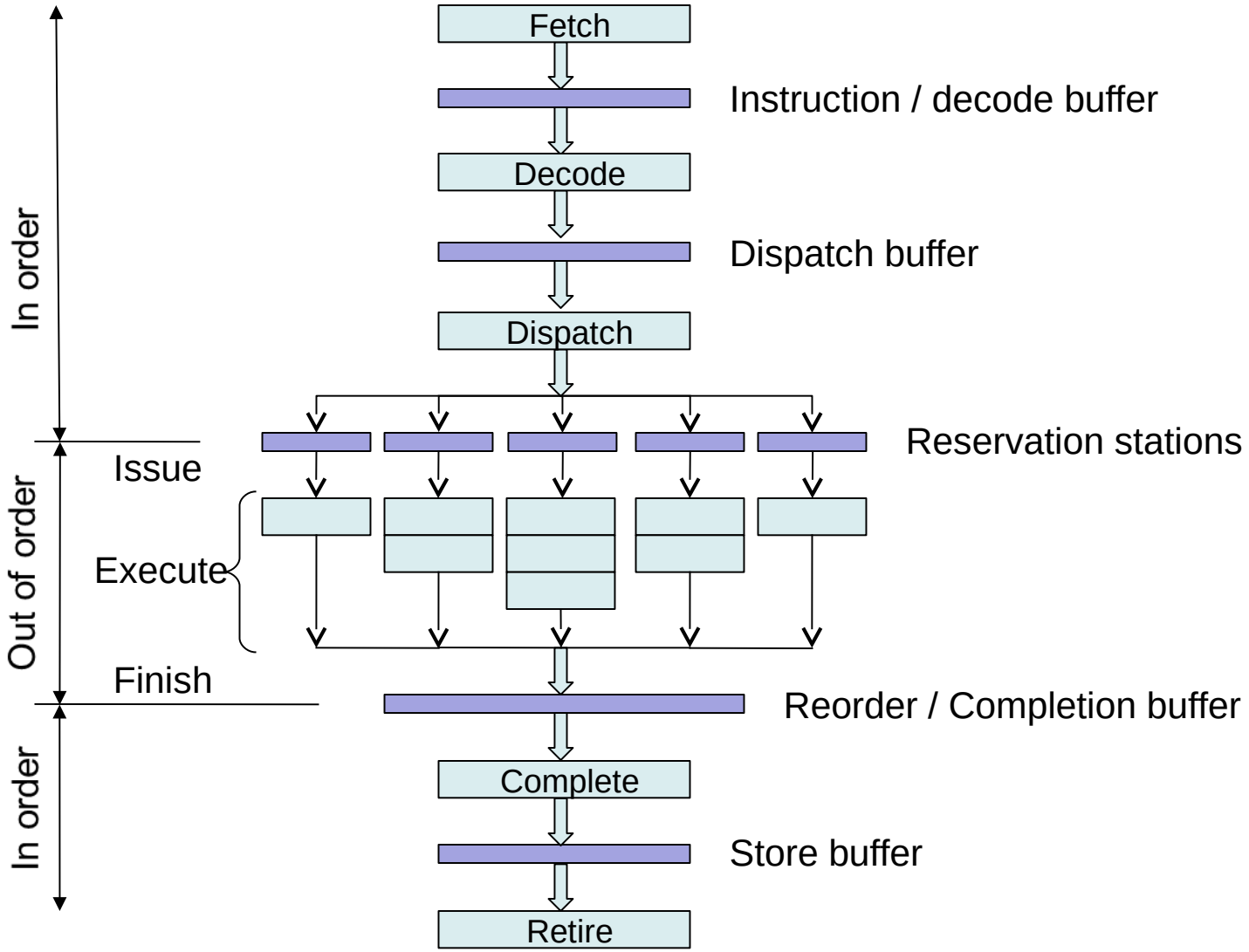
Pipelined execution, branches cause even more problems

- Early branch and jump instructions decode
- Processes instructions in delay slots MIPS, DSP
- Static and dynamic branch prediction, branch target address buffer (cache), speculative instructions execution



Source: wikipedia

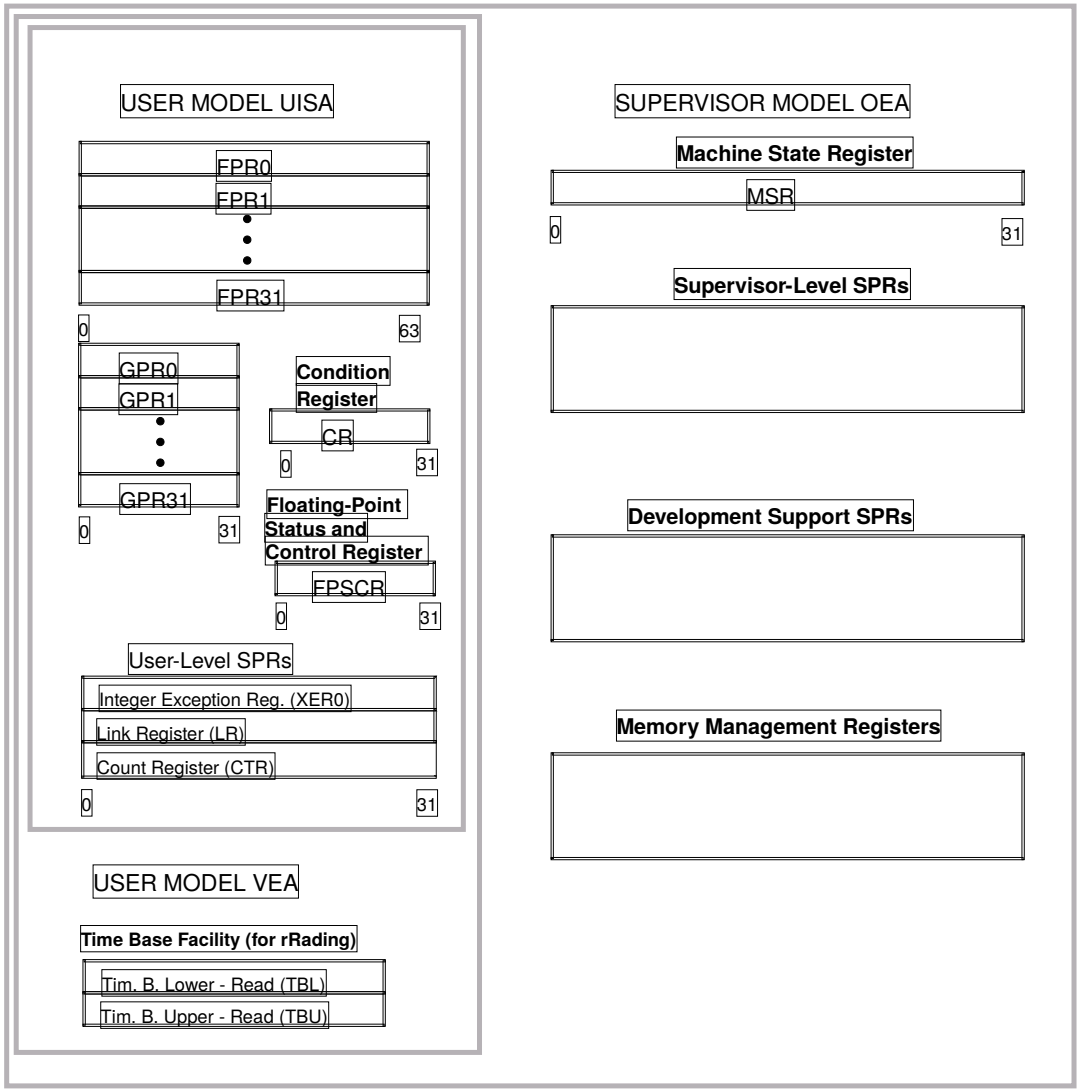
Parallel Instructions Execution – Superscalar CPU



Other techniques to reduce memory access frequency ⇒ register windows, link/return address register

- Generally more registers (RISCs usually 31+1, ARM 32-bit 16, compare with 32-bit 386 still only 8 registers)
- SPARC - 8 global registers, 8 from previous window (parameters), 16 in actual window, up to 100 and more registers to stack windows. 8 registers in actual window is used to pass parameters into subroutine
- PowerPC, MIPS, ARM – speedup to call leaf-node functions with use of return address (link register) to store address of the instruction to be executed after return from subroutine

PowerPC Architecture



Summit Supercomputer – IBM AC922 – 2018 TOP500 #1

- June 2018, US Oak Ridge National Laboratory (ORNL), 200 PetaFLOPS, 4600 “nodes”, 2× IBM Power9 CPU +
- 6× Nvidia Volta GV100
- 96 lanes of PCIe 4.0, 400Gb/s
- NVLink 2.0, 100GB/s CPU2GPU
- GPU-to-GPU
- 2TB DDR4-2666 per node
- 1.6 TB NV RAM per node
- 250 PB storage

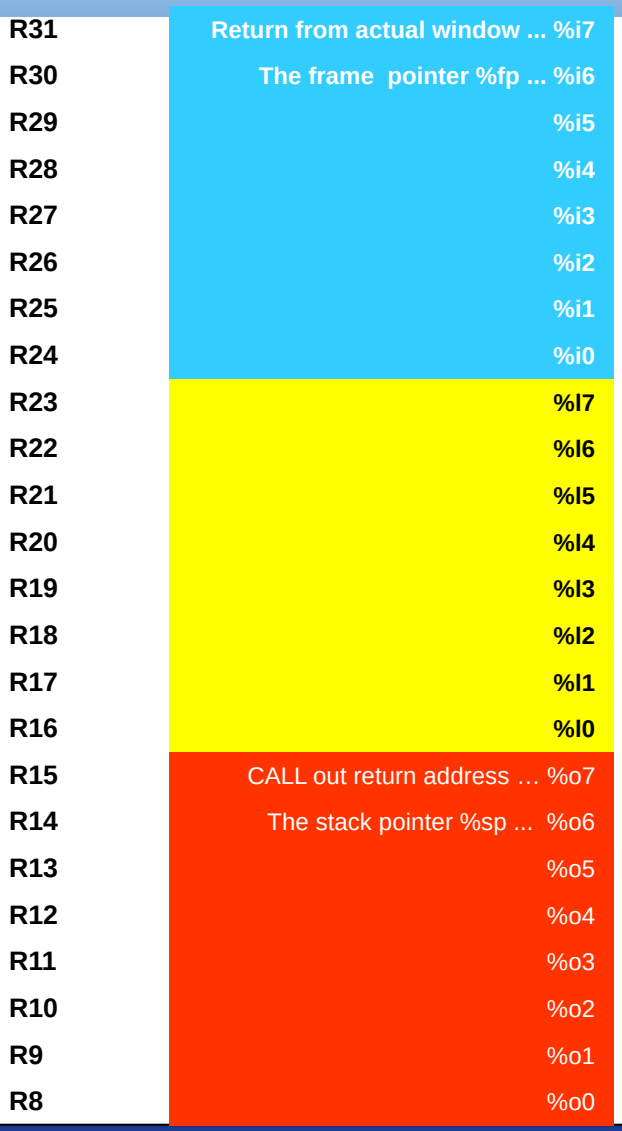
Source: http://www.tomshardware.com/

- POWER9-SO, Global Foundries 14nm FinFET, 8×10^9 tran., 17-layer, 24 cores, 96 threads (SMT4)
- 120MB L3 eDRAM (2 CPU 10MB), 256GB/sv

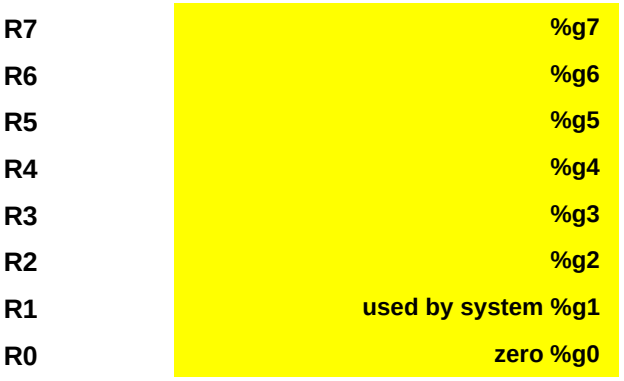
SPARC – Register Windows

- CPU includes from 40 to 520 general purpose 32-bit registers
- 8 of them are global registers, remaining registers are divided in groups of 16 into at least 2 (max 32) register windows
- Each instruction has access to 8 global registers and 24 registers accessible through actually selected register windows position
- 24 windowed registers are divided into 8 input (in), 8 local (local) and 8 registers from the following window which are visible through current window as an output (out) registers (registers to prepare call arguments)
- Active window is given by value of 5-bit pointer – Current Window Pointer (CWP).
- CWP is decremented when subroutine is entered which selects following window as an active/current one
- Increment of CWP return to the previous register window
- Window Invalid Mask (WIM) is a bit-map which allows to mark any of windows as invalid and request exception (overflow or underflow) when window is activated/selected by CWP

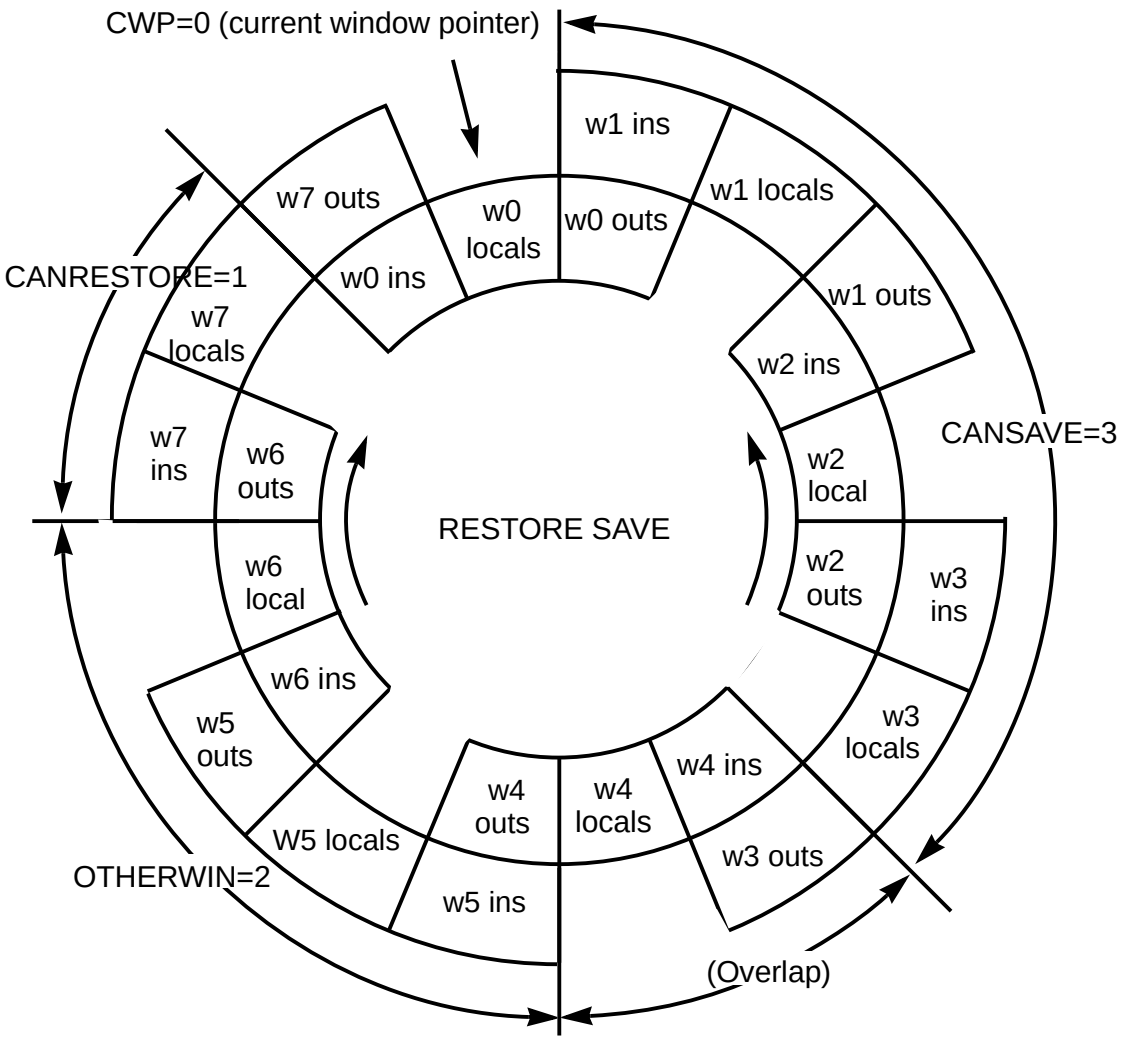
SPARC - Registers



G (global)

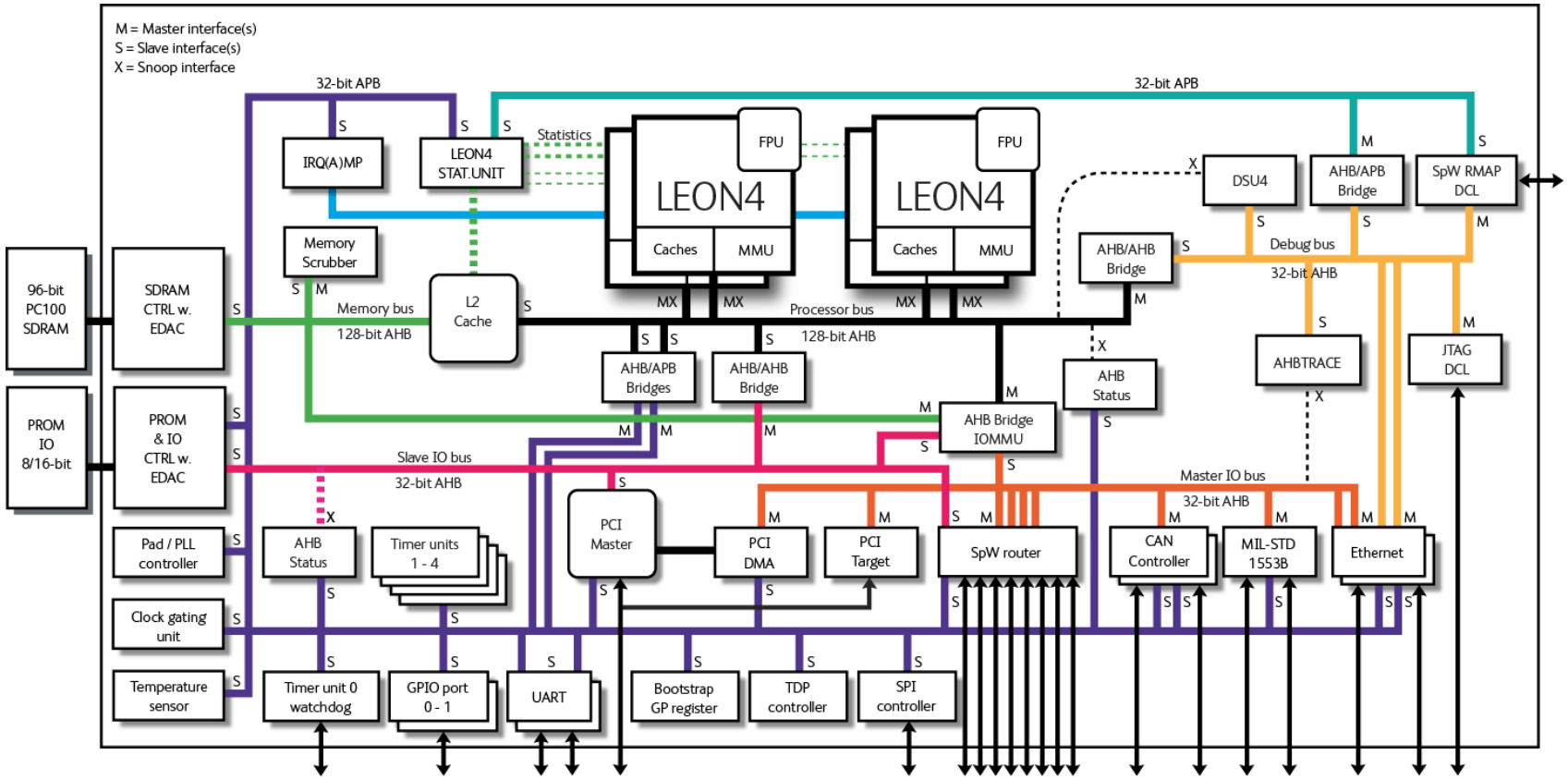


SPARC – Register Windows Operation



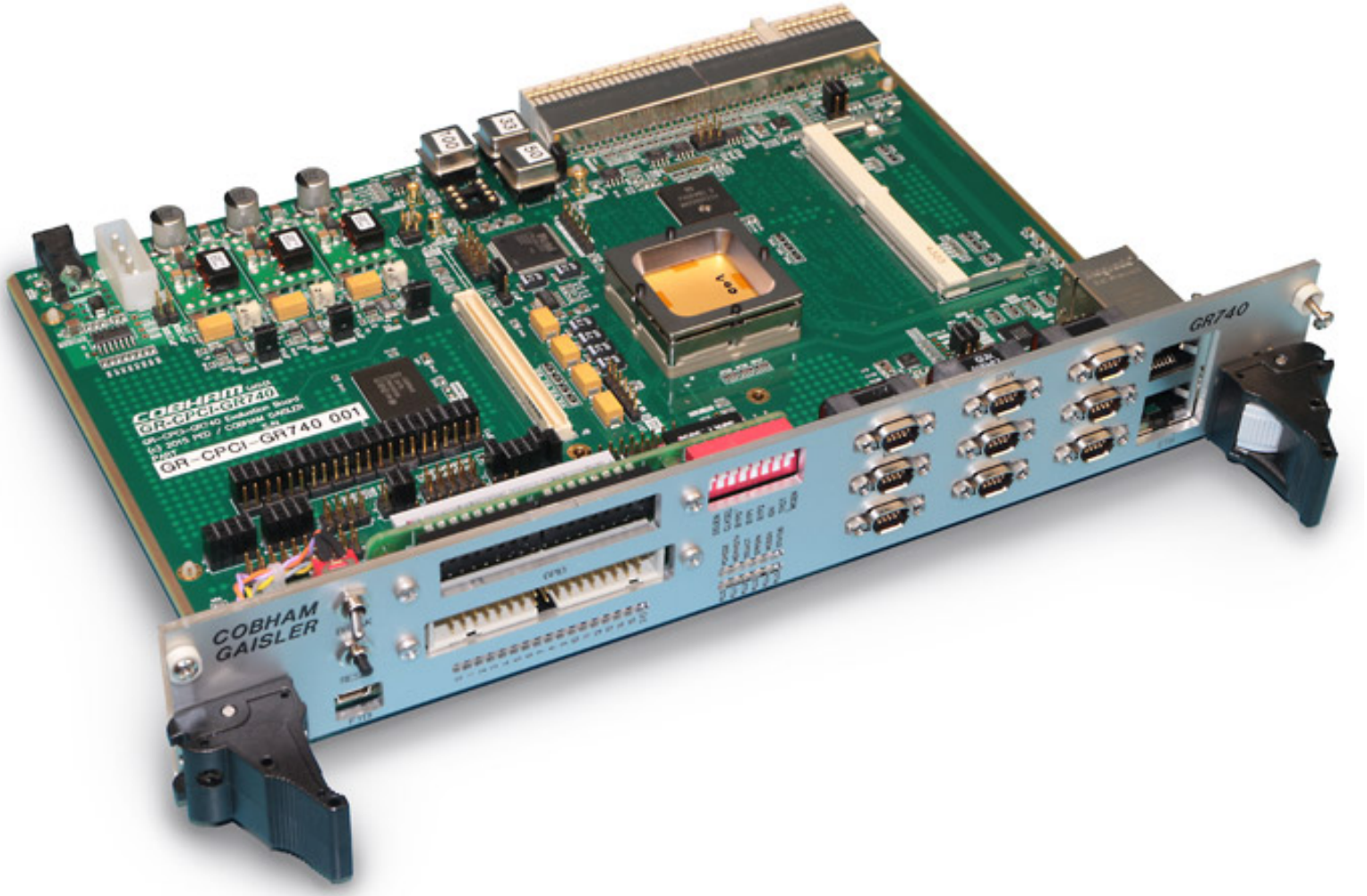
SPARC in Space – Cobham Gaisler GR740

- Fault-tolerant, quad-core, SPARC V8, 7-stage pipeline, 8 register windows, 4x4 KiB I + 4x4 KiB D cache, IEEE-754, 2 MiB L2 cache



Source: https://www.gaisler.com/index.php/products/components/gr740

Quad-Core LEON4FT (GR740) Development Board



Source: https://www.gaisler.com/index.php/products/boards/gr-cpci-gr740

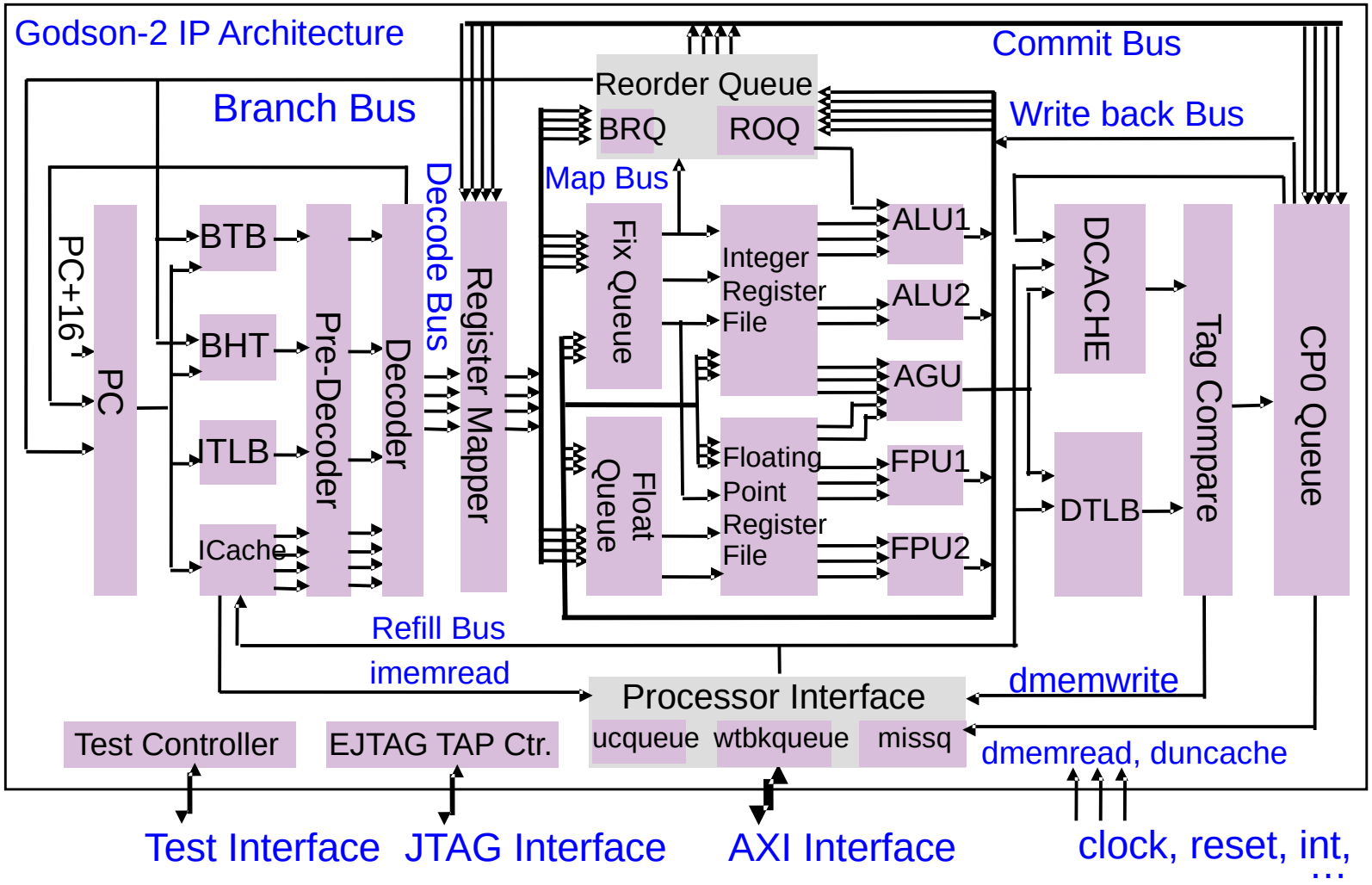
MIPS Architecture Variants

- Probably architecture with the highest use count on Earth at one time (all kinds of AP, embedded systems, etc.)
- Development still continues even for high performance desktops and supercomputers use – Loongson3A
- MIPS Aptiv – MIPS32 MCU for embedded applications
- MIPS Warrior – MIPS P6600 MIPS64 Release 6 – hardware virtualization with hardware table walk, 128-bit SIMD
- MIPS architecture inspired many SoftCore designs for FPGA, examples
  - Xilinx Microblaze
  - Altera Nios

Pipelined execution, no microcode, but still problems with jump instructions

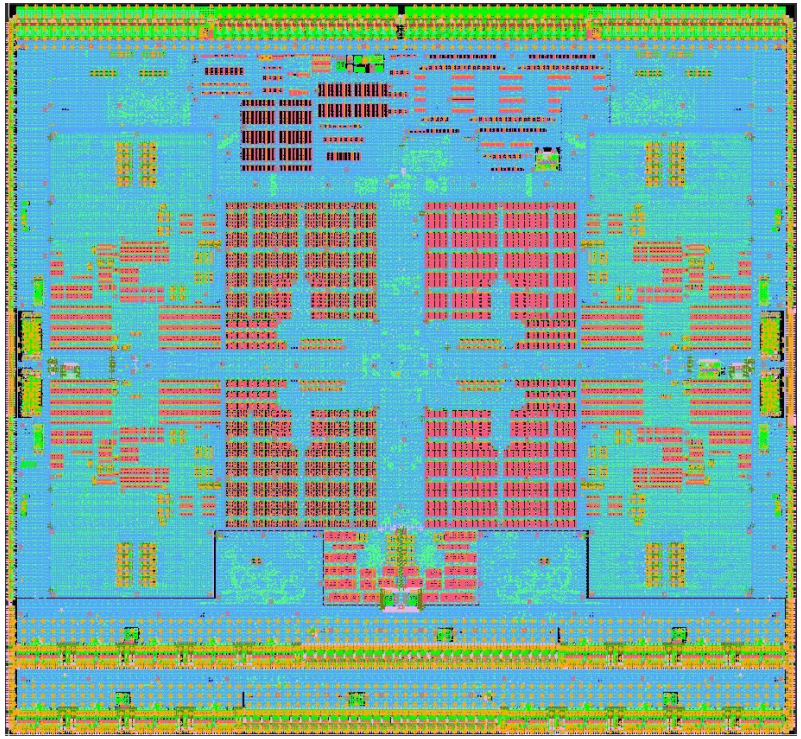
- Early jump instruction decode
- Use delay slots to keep pipeline busy, MIPS, DSP
- Static and dynamic conditional branch prediction, branch target address cache, speculative instruction execution

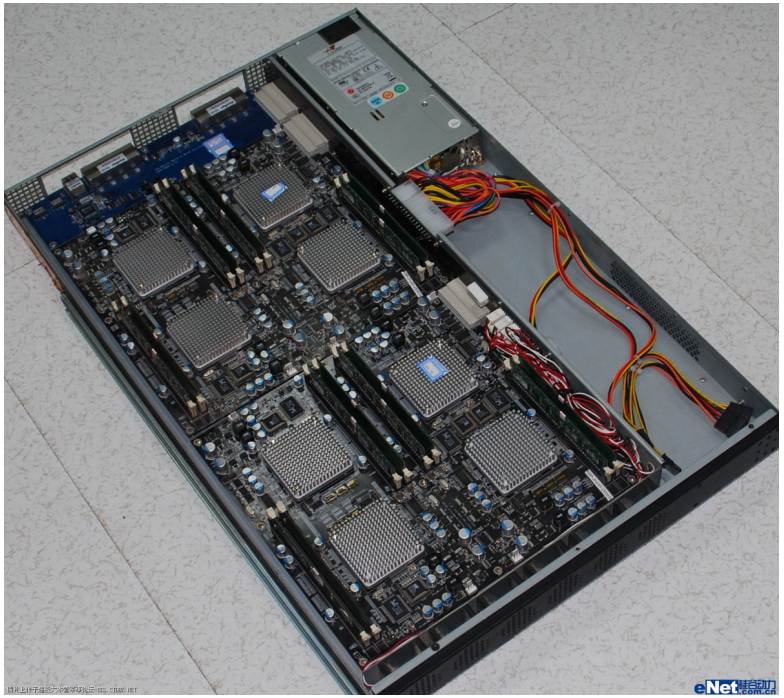
Loongson3A


Godson-2 IP Architecture
Commit Bus
Reorder Queue
BRQ
ROQ
Branch Bus
Write back Bus
Map Bus
Decode Bus
PC+16
PC
BTB
BHT
JTLB
ICache
Pre-Decoder
Decoder
Register Mapper
Fix Queue
Float Queue
Integer Register File
Floating Point Register File
ALU1
ALU2
AGU
FPU1
FPU2
DCACHE
DTLB
Tag Compare
CP0 Queue
Refill Bus
imemread
Processor Interface
dmemwrite
ucqueue
wtbkqueue
missq
dmemread, duncache
Test Controller
EJTAG TAP Ctr.
Test Interface
JTAG Interface
AXI Interface
clock, reset, int, …

MIPS 64-bit Base of China Computing

- Loongson 3A5000 – desktop processor, 12nm, 4 cores @ at 2.5GHz
- Loongson 3C5000 – server processor, 12nm, 16 cores, supports 4 to 16-way servers



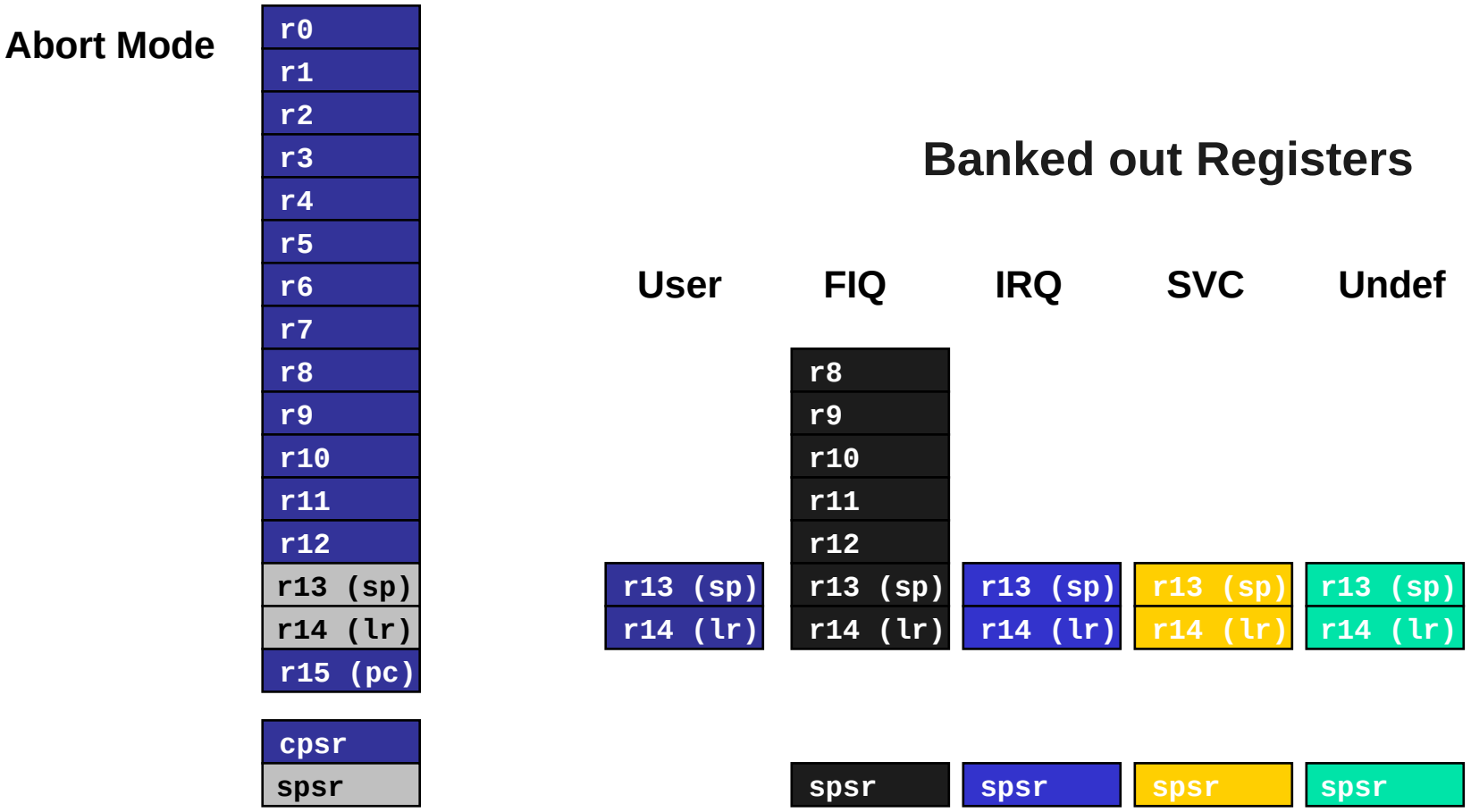


Attempts to enhance code density ⇒ shorter aliases, variable instruction length even for RISC, VLIW

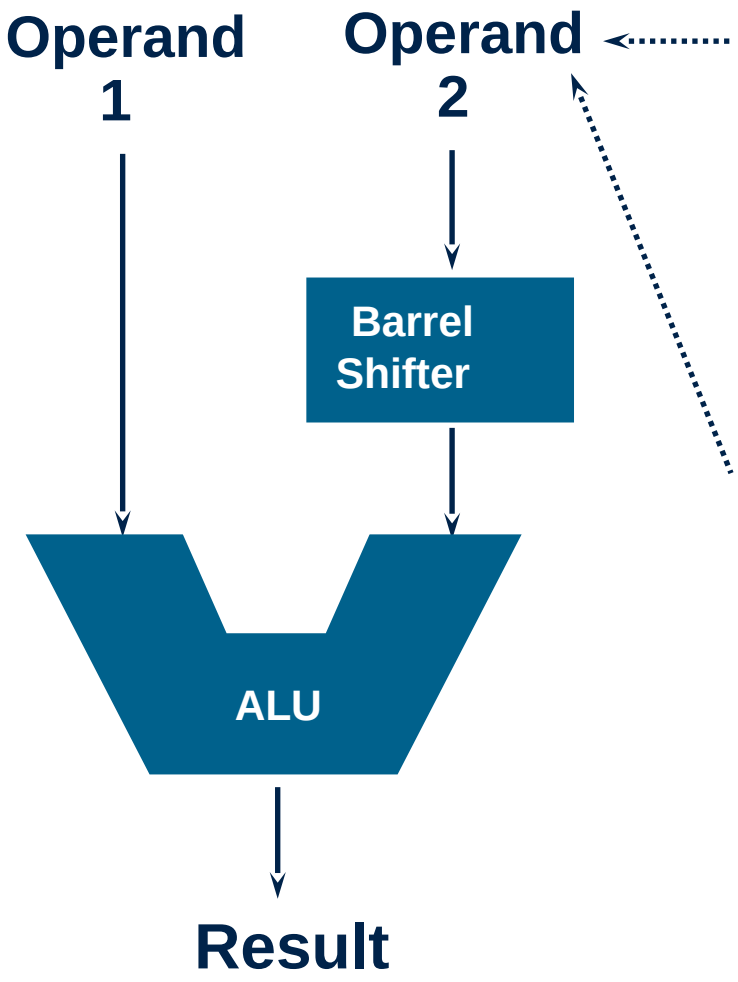
- ARM, 16bit aliases for most common 32bit instructions (Thumb mode, requires mode switching, later on function by function call basis)
- MIPS Aptiv, same on function basis
- M-Core, 32-bit CPU but only 16-bit instruction encoding
- SuperH, 32/64-bit CPU, 16-bit instructions encoding
- ColdFire - RISC implementation based on 68000 instruction set, but only 16, 32, 48-bit length instructions are accepted
- RISC-V, reserved bits of 32-bit instructions to allow seamless combination of 32 and 16-bit instructions

ARM Architecture - Registers

Current Visible Registers



ARM Architecture – ALU and Operands Encoding



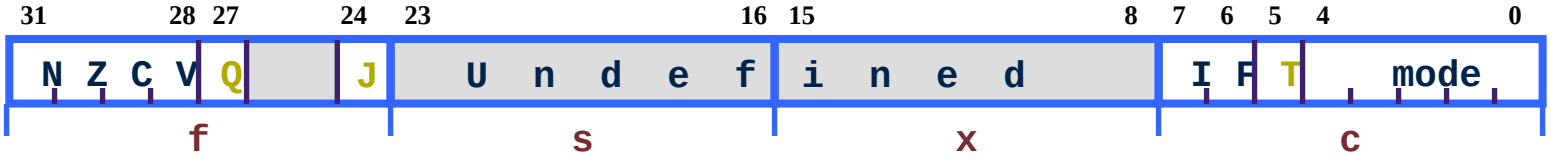
Register, optionally with shift operation

- Shift value can be either be:
  - 5 bit unsigned integer
  - Specified in bottom byte of another register.
- Used for multiplication by constant

Immediate value

- 8 bit number, with a range of 0-255.
  - Rotated right through even number of positions
- Allows increased range of 32-bit constants to be loaded directly into registers

ARM Architecture – Program Status Word



- Condition code flags
  - N = Negative result from ALU
  - Z = Zero result from ALU
  - C = ALU operation Carried out
  - V = ALU operation oVerflowed
- Sticky Overflow flag - Q flag
  - Architecture 5TE/J only
  - Indicates if saturation has occurred
- J bit
  - Architecture 5TEJ only
  - J = 1: Processor in Jazelle state
- Interrupt Disable bits.
  - I = 1: Disables the IRQ.
  - F = 1: Disables the FIQ.
- T Bit
  - Architecture xT only
  - T = 0: Processor in ARM state
  - T = 1: Processor in Thumb state
- Mode bits
  - Specify the processor mode

ARM Architecture – CPU Execution Modes

- User : unprivileged mode under which most tasks run
- FIQ : entered when a high priority (fast) interrupt is raised
- IRQ : entered when a low priority (normal) interrupt is raised
- Supervisor : entered on reset and when a Software Interrupt instruction is executed
- Abort : used to handle memory access violations
- Undef : used to handle undefined instructions
- System : privileged mode using the same registers as user mode

Conclusion – Almost

- There is no magic solution for all discussed problems for all use cases
- It is necessary to combine discussed techniques and optimize the mix according to intended CPU area of use (the highest computational power/power efficient)
- use of heterogeneous systems for high performance computation – vector units, GPU, FPGA accelerators

Why Instruction Set Architecture Matters

- Why can't Intel sell mobile chips?

  99%+ of mobile phones/tablets are based on ARM's v7/v8 ISA

- Why can't ARM partners sell servers?

  99%+ of laptops/desktops/servers are based on the AMD64 ISA (over 95%+ built by Intel)

- How can IBM still sell mainframes?

  IBM 360 is the oldest surviving ISA (50+ years)

ISA is the most important interface in a computer system

ISA is where software meets hardware. (SiFive/RISC-V)

ARM 64-bit – AArch64

- Calling uses LR, no register banking, ELR for exceptions
- PC is separate register (not included in general purpose registers file)
- 31 64-bi registers R0 to R30 (R30 = X30 ≅ LR)
  - Symbol W*n* (W0) used for 32-bit access, X*n* (X0) for 64-bit
  - Reg. code 31 same zero role as MIPS 0, WZR/XZR in code
  - Reg. code 31 special meaning as WSP, SP for some opcodes
- Immediate operand 12-bit with optional LS 12 for arithmetics operations and repetitive bit masks generator for logic ones
- 32-bit operations ignores bits 32–63 for source and zeros these in the destination register

AArch64 – Branches and Conditional Operations

- Omitted conditional execution in all instructions as well as Thumb IT mechanism
- Conditional register retain, CBNZ, CBZ, TBNZ, TBZ added
- Only couple of conditional instructions
  - add and sub with carry, select (move C?A:B)
  - set 0 and 1 (or -1) according to the condition evaluation
  - conditional compare instruction
- 32-bit and 64-bit multiply and divide (3 registers), multiply with addition $64\times64+64 \rightarrow 64$ (four registers), high bits 64 to 127 from 64×64 multiplication

AArch64 – Memory Access

- 48+1 bit address, sign extended to 64 bits
- Immediate offset can be multiplied by access size optionally
- If register is used in index role, it can be multiplied by access size and can be limited to 32 bits
- PC relative ±4GB can be encoded in 2 instructions
- Only pair of two independent registers LDP and STP (ommited LDM, STM), added LDNP, STNP
- Unaligned access support
- LDX/STX(RBHP) for 1,2,4,8 and 16 bytes exclusive access

AArch64 – Address Modes

- Simple register (exclusive)

  [base{,#0}]

- Offset

  [base{,#imm}] – Immediate Offset

  [base,Xm{,LSL #imm}] – Register Offset

  [base,Wm,(S|U)XTW {#imm}] – Extended Register Offset

- Pre-indexed

  [base,#imm]!

- Post-indexed

  [base],#imm

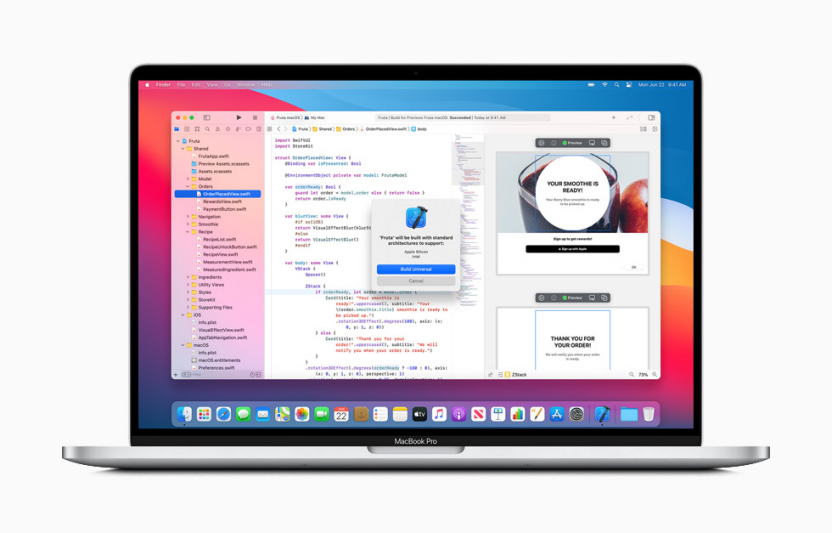
- PC-relative (literal) load

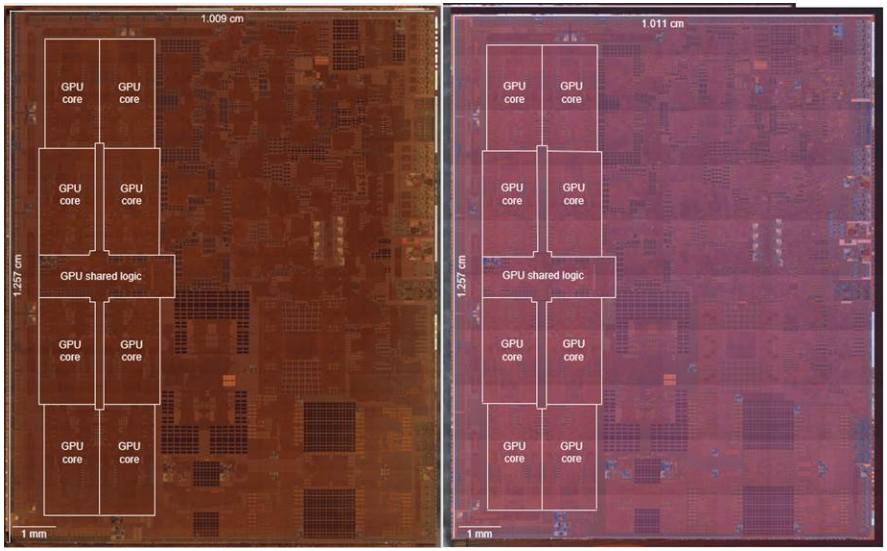
  label

Bits	Sign	Scaling	WBctr	LD/ST type
0	-	-	-	LDX, STX, acquire, release
9	signed	scaled	option	reg. pair
10	signed	unscaled	option	single reg.
12	unsig.	scaled	no	single reg.

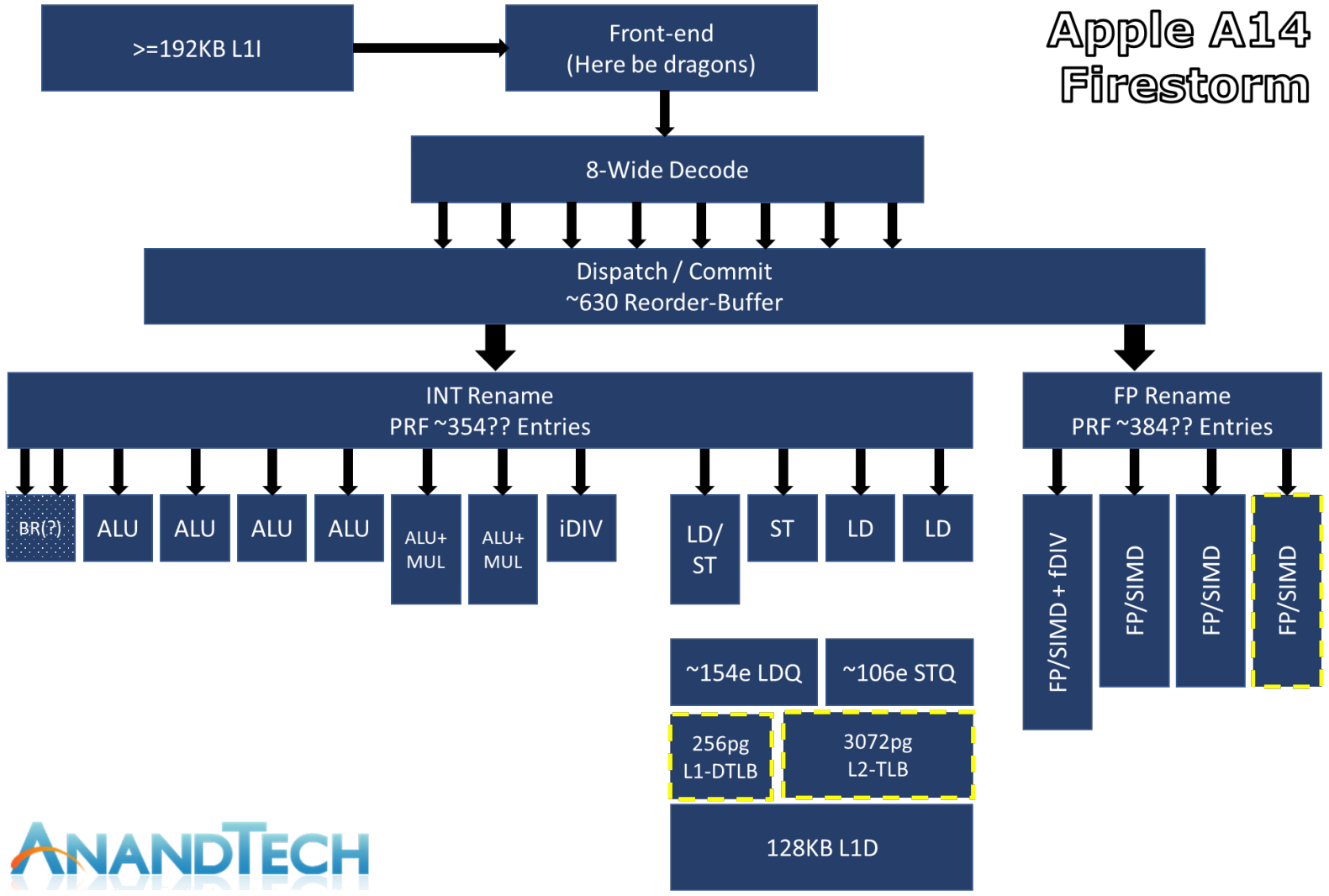
Apple A12Z Bionic – 64-bit ARM-based

- People who are really serious about software should make their own hardware. *Alan Kay*
- *Apple A12Z, 8 cores (ARM big.LITTLE: 4 "big" Vortex + 4 "little" Tempest), Max. 2.49 GHz, ARMv8.3-A*
- *Cache L1 128 KB instruction, 128 KB datam L2 8 MB*
- *GPU Apple designed 8-Core*





Apple M1, A14, 4 Firestorm, 4 Icestorm



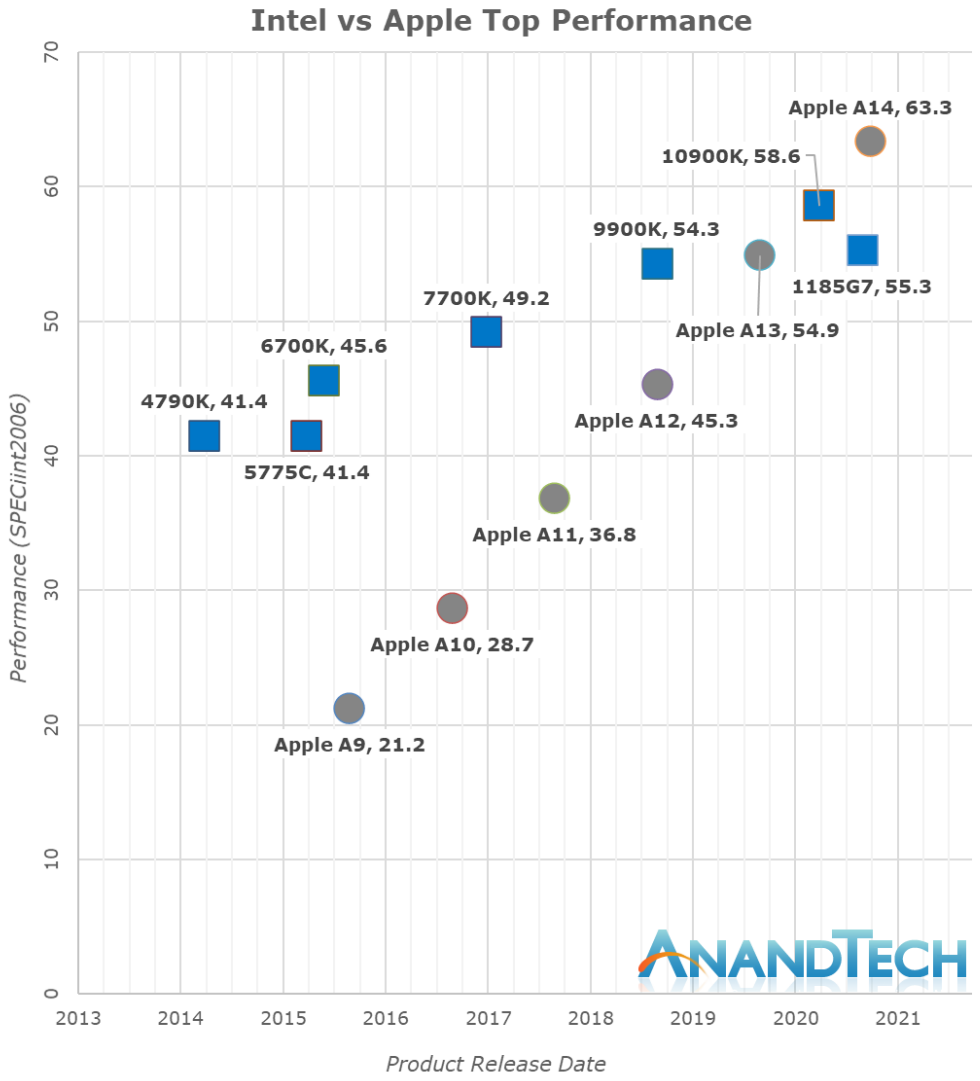
Source: https://www.anandtech.com/show/16226/apple-silicon-m1-a14-deep-dive

Intel versus Apple Top Single Thread Performance

Last 5 years

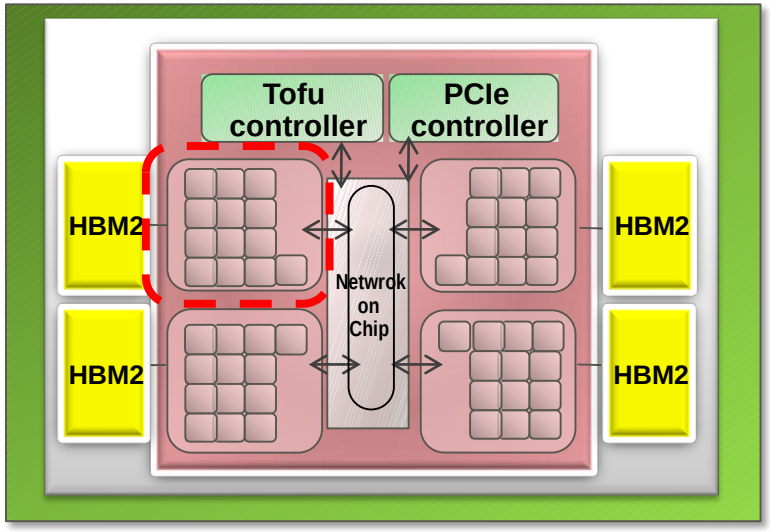
Intel +28%

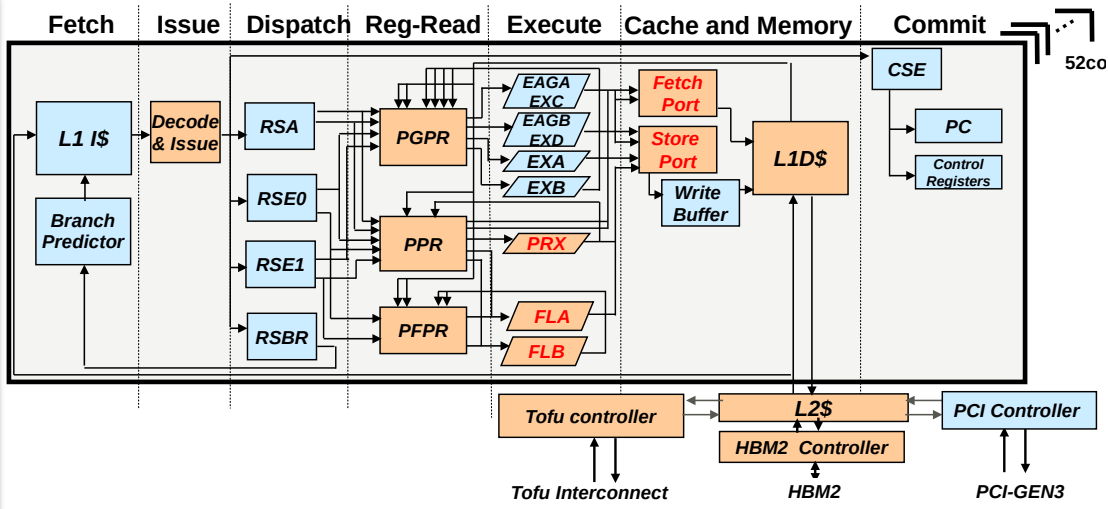
Apple +198%
2.98×



Fujitsu – Supercomputer Fugaku – A64FX, 2020 TOP500 #1

- Combine Armv8.2-A (AArch64 only) with Fujistu supercomputer technology, SPARC64 V till now
- 48 computing cores + 4 assistant cores, SVE 512-bit wide SIMD
- HBM2 32GiB, 7nm FinFET, 8,786M transistors
- Tofu 6D Mesh/Torus, 28Gbps x 2 lanes x 10 ports, PCIe

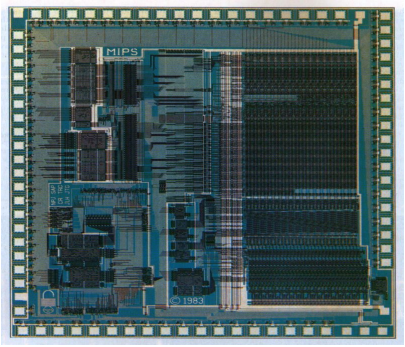




Source: Fujitsu High Performance CPU for the Post-K Computer, 2018

CISC and RISC Origin and Lesson Learned Again

- IBM PC 1981 picks Intel 8088 for 8-bit bus (and Motorola 68000 out of main business)
- Use SRAM for instruction cache of user-visible instructions
- Use simple ISA – Instructions as simple as microinstructions, but not as wide, Compiled code only used a few CISC instructions anyways, Enable pipelined implementations
- CISC executes fewer instructions per program (≈ 3/4X instructions), but many more clock cycles per instruction (≈ 6X CPI)
- ⇒ RISC ≈ 4X faster than CISC
- Chaitin's register allocation scheme benefits load-store ISAs
- Berkeley (RISC I, II → SPARC) & Stanford RISC Chips (MIPS)

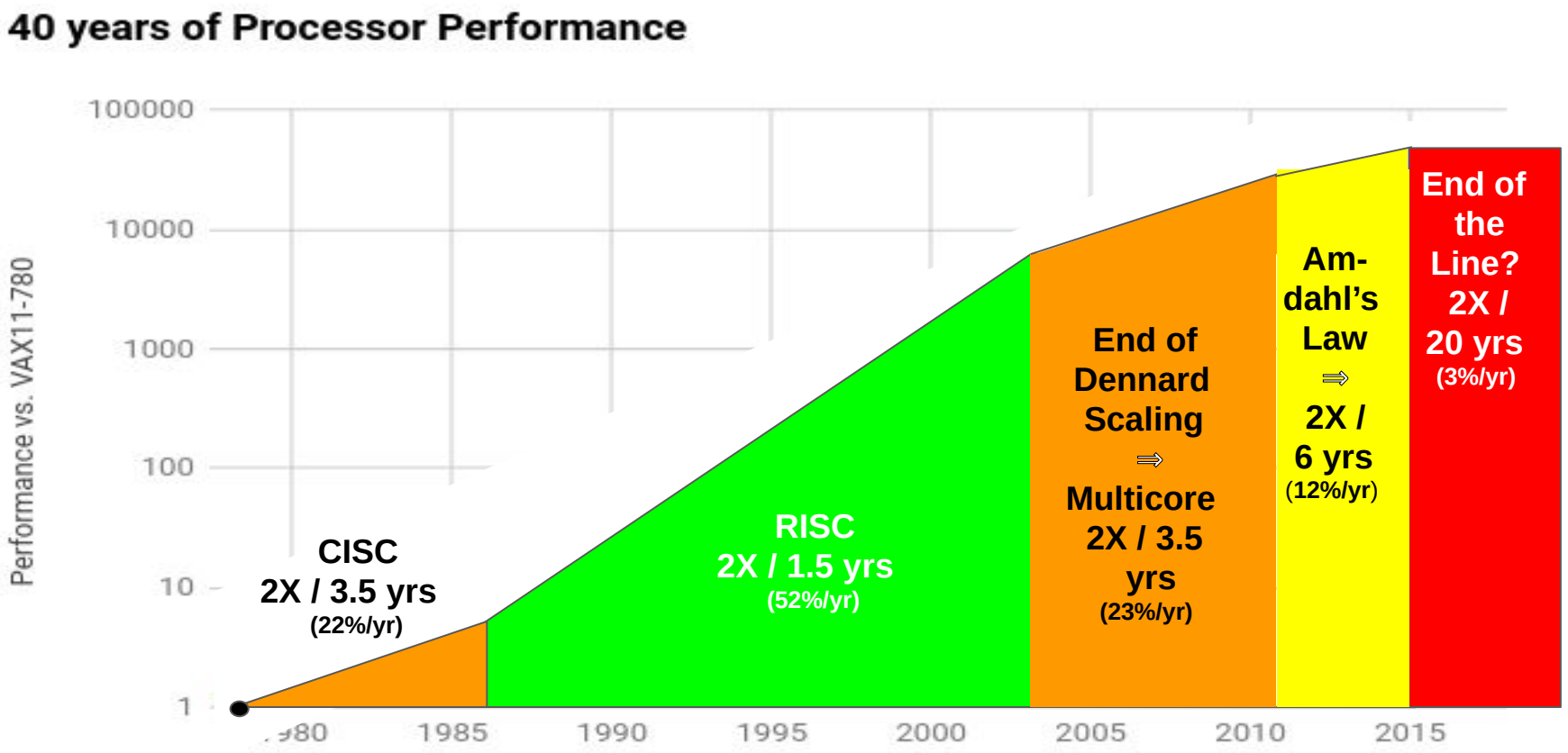


Referenced again to refresh talk from the first – introductory lesson

Stanford MIPS (1983) contains 25,000 transistors, was fabbed in 3 µm &4 µm NMOS, ran at 4 MHz (3 µm) and size is 50 mm2 (4 µm) (Microprocessor without Interlocked Pipeline Stages)

Source: A New Golden Age for Computer Architecture with prof. Patterson permission

End of Growth of Single Program Speed?



Based on SPECintCPU. Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018

RISC-V – Optimize and Simplify RISC Again

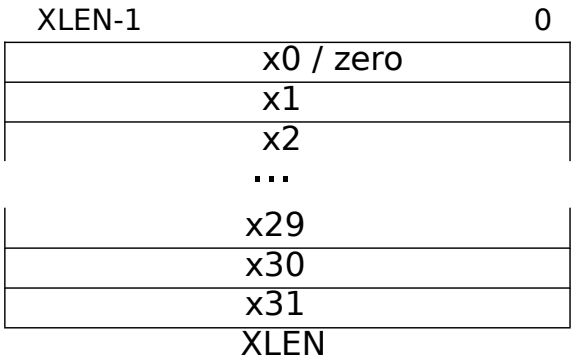
- Patterson, Berkeley RISC 1984 → initiation of RISC era, evolved into **SPARC** (Hennessy **MIPS**, Stanford University)
- Commercialization and extensions results in too complex CPUs again, with license and patents preventing even original inverstors to use real/actual implementations in silicon to be used for education and research
- MIPS is model architecture for prevalent amount of base courses and implementation of similar processor is part of follow up courses (A4M36PAP)
- Krste Asanovic and other Dr. Patterson's students initiated development of new architecture (start of 2010)
- BSD Licence to ensure openness in future
- Supported by GCC, binutils., Linux, QEMU, etc.
- Simpler than SPAC, more like MIPS but optimized on gate level load (fanout) and critical paths lengths in future designs
- Some open implementations already exists **Rocket** (SiFive, BOOM), project **lowRISC** contributes to research in security area, in ČR Codasip
- Already more than 20 implementations in silicon

RISC-V – Architecture Specification

- ISA specification can be found at http://riscv.org/
  - The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.0
  - Andrew Waterman, Yunsup Lee, David Patterson, Krste Asanovic
  - Not only architecture description but even choices analysis with pro&cons of each selection and cost source description/analysis of alternatives
- classic design, 32 integer registers, the first tied to zero, regsrc1, regsrc2, regdest operands, uniqueness, rule kept strictly even for SaveWord, leads to non-continuous immediate operands encoding, PC not part of base register file, PC-relative addressing
- variants for 32, 64 a 128-bit registers and address-space defined
- high code density (16-bit instruction encoding variant planned)
- encoding reserves space for floating point (single, double, quad) and multimedia SIMD instructions systematic way, etc.

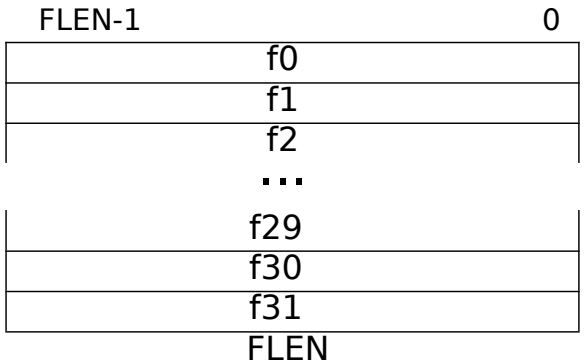
RISC-V – Registers

Integer registers



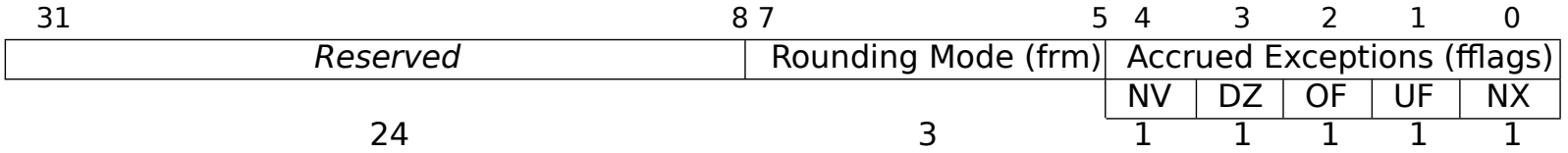
Variant	XLEN
RV32	32
RV64	64
RV128	128

Floating point registers



Variant	FLEN
F	32
D	64
Q	128

Floating-point control and status register

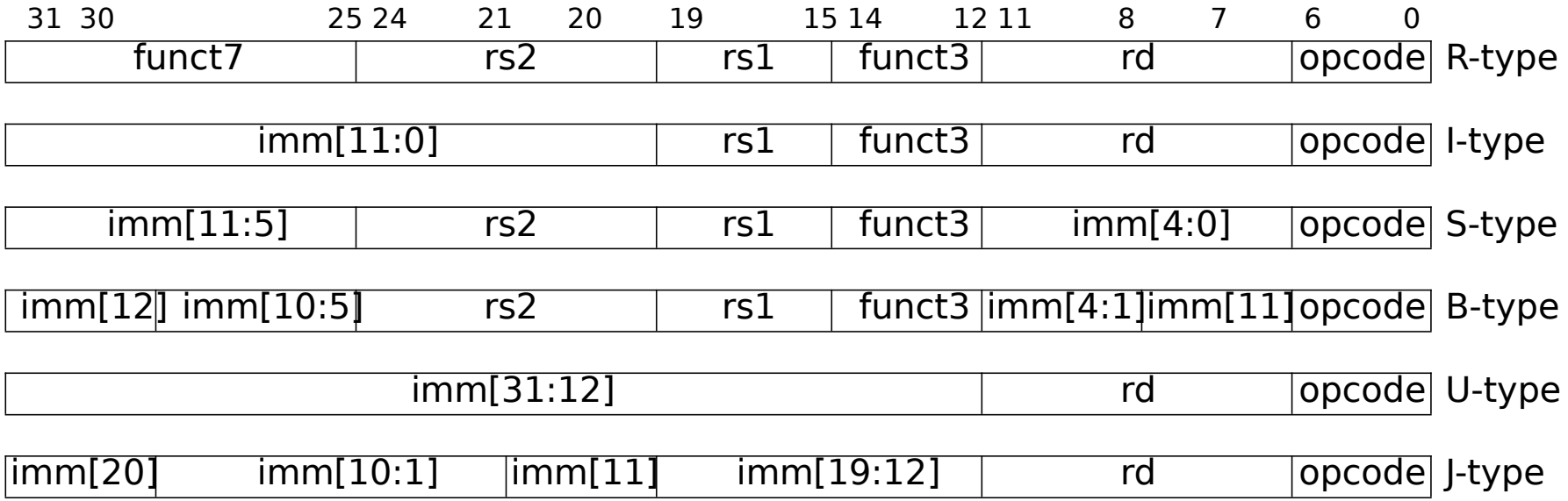


Source: https://riscv.org/specifications/

RISC-V – Instruction Length Encoding

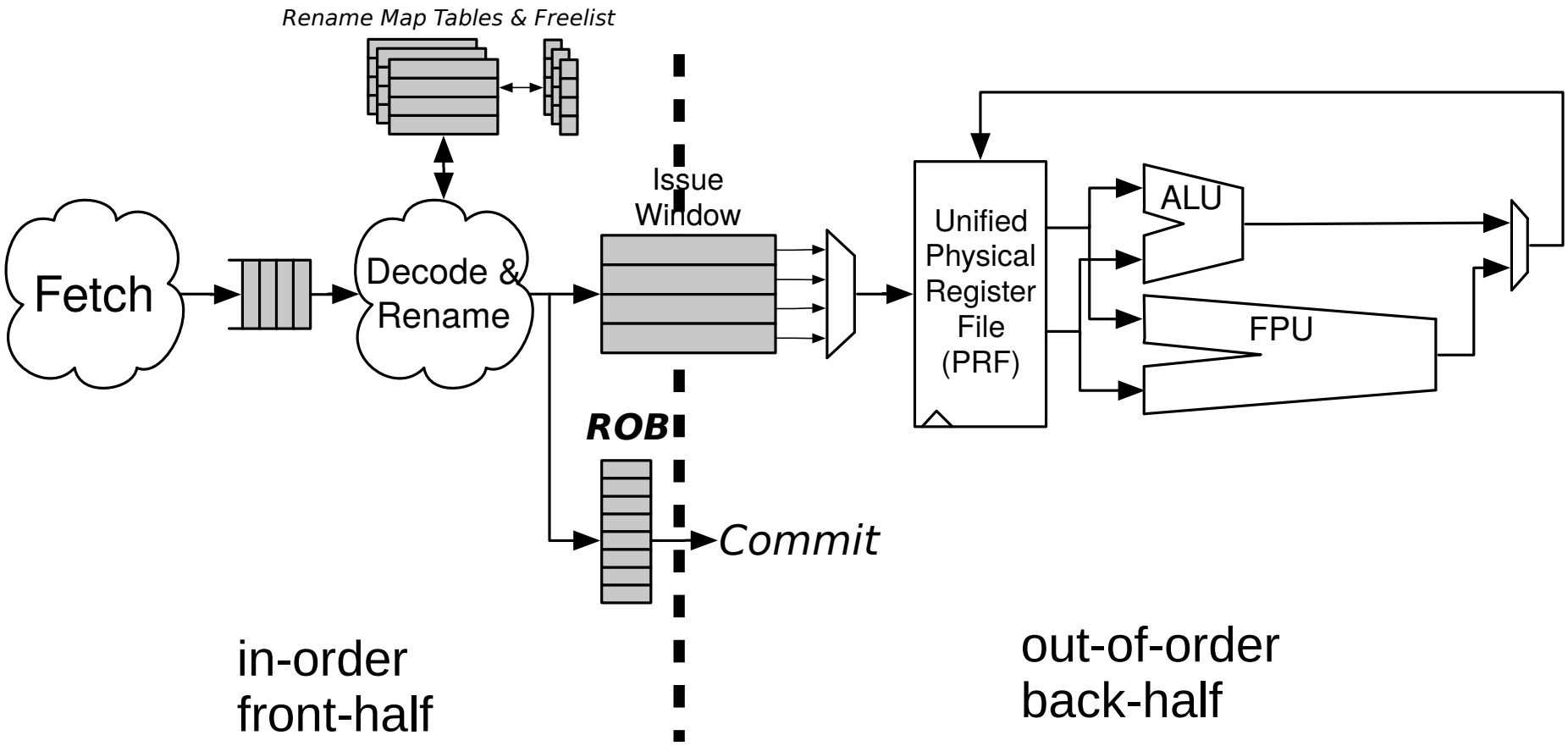
Source: https://riscv.org/specifications/

RISC-V – 32-bit Instructions Encoding



Source: https://riscv.org/specifications/

BOOM Superscalar RISC-V into Rocket Chip

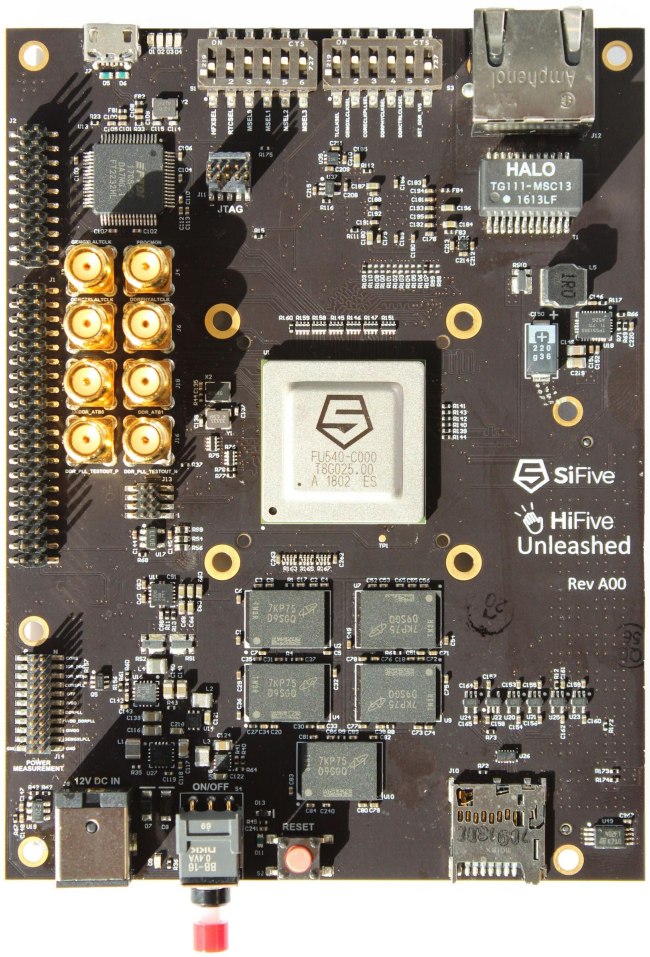


Main developer: Christopher Celio

9k source lines + 11k from Rocket

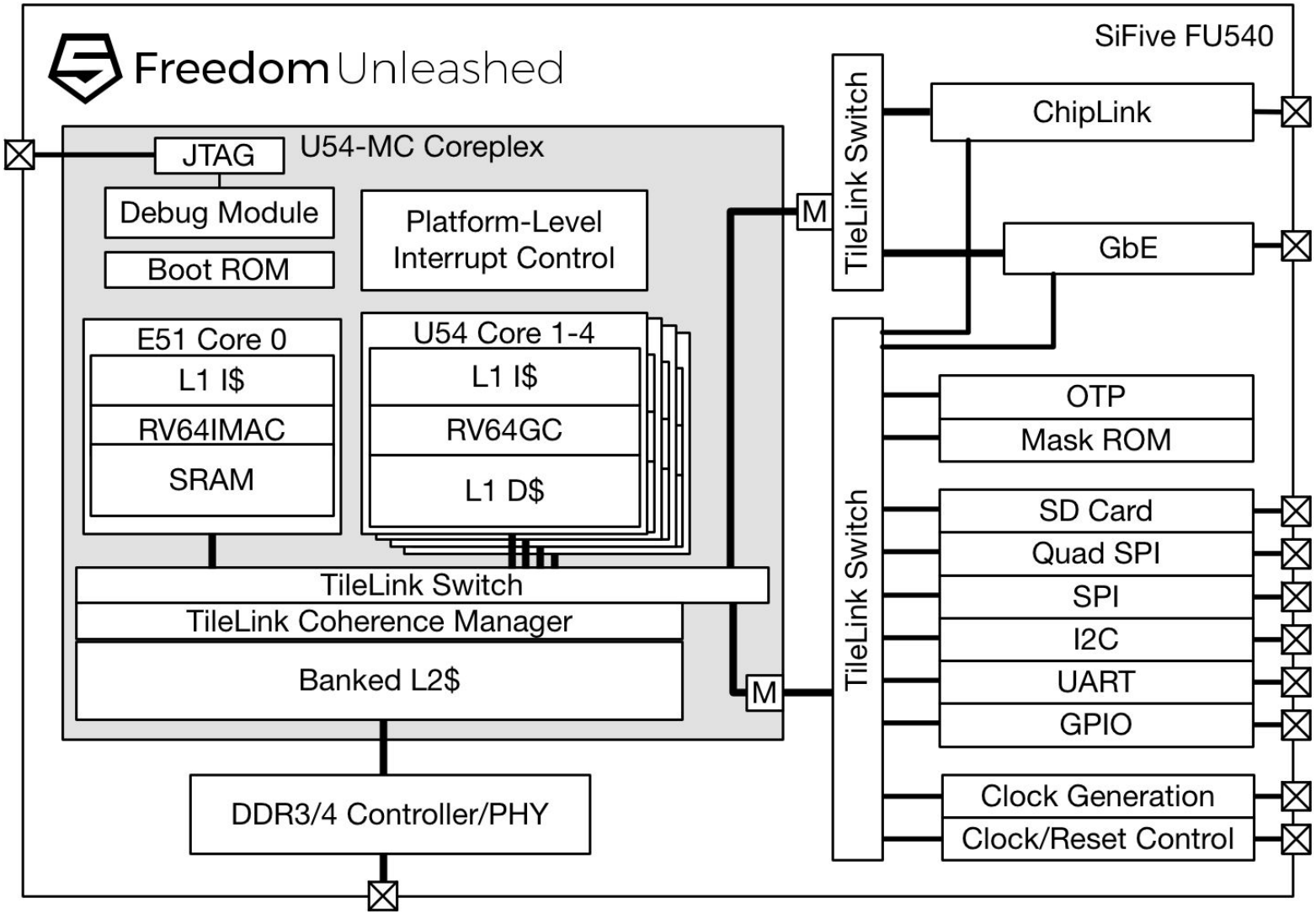
Source: https://riscv.org/wp-content/uploads/2016/01/Wed1345-RISCV-Workshop-3-BOOM.pdf

RISC-V – HiFive Unleashed

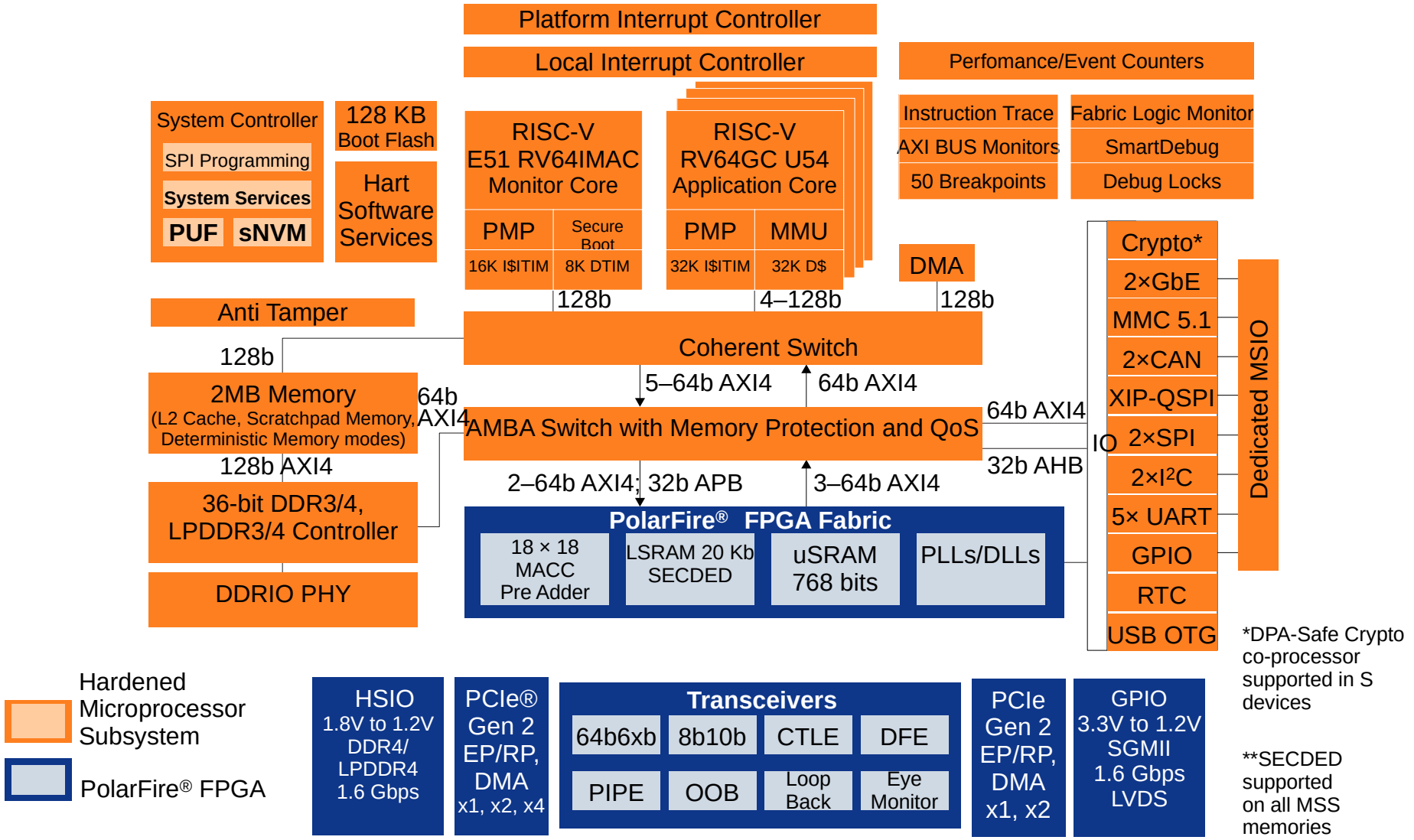


- SiFive FU540-C000 (built in 28nm)
  - 4+1 Multi-Core Coherent Configuration, up to 1.5 GHz
  - 4x U54 RV64GC Application Cores with Sv39 Virtual
  - Memory Support
  - 1x E51 RV64IMAC Management Core
  - Coherent 2MB L2 Cache
  - 64-bit DDR4 with ECC
  - 1x Gigabit Ethernet
- 8 GB 64-bit DDR4 with ECC
- Gigabit Ethernet Port
- 32 MB Quad SPI Flash
- MicroSD card for removable storage
- FMC connector for future expansion with add-in card

RISC-V – HiFive Unleashed



Microchip PolarFire® SoC+FPGA



BeagleV

- StarLight JH7100
- StarFive RISC-V U74 @1.0 GHz dual-core 64-bit RV64GC ISA
- 4GB/ 8GB LPDDR4, USB 3.0 ports, 40 pin GPIO
- Vision DSP Tensilica-VP6 @ 600MHz
- NVDLA Engine (2048 MACs @ 800MHz)
- Neural Network Engine (1024MACs @ 500MHz)
- https://beagleboard.org/beaglev
- https://www.sifive.com/cores/u74-mc

More RISC-V projects

- Libre RISC-V https://libre-riscv.org/
  - Quad-core 28nm RISC-V 64-bit (RISCV64GC core with Vector SIMD Media / 3D extensions)
  - 300-pin 15x15mm BGA 0.8mm pitch
  - 32-bit DDR3/DDR3L/LPDDR3 memory interface
- Espressif ESP32-C3 single core (FreeRTOS fork or NuttX)
- NOEL-V RISC-V Processor - Cobham Gaisler
- More RISC-V resources
  - https://riscv.org/
  - RISC-V YouTube channel https://www.youtube.com/channel/UC5gLmcFuvdGbajs4VL-WU3g