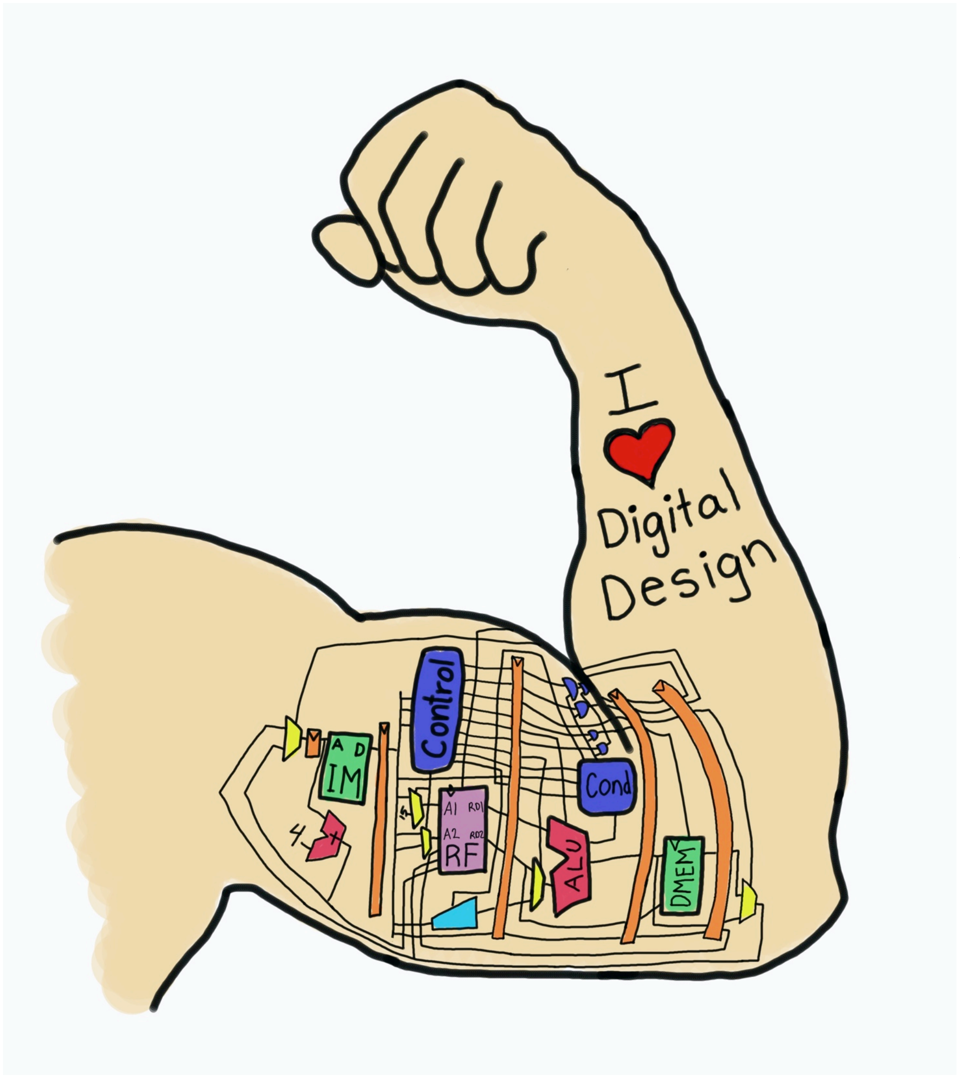
E85 Digital Design & Computer Engineering



Lecture 24:

Evolution of ARM Processors

HARVEY MUDD COLLEGE

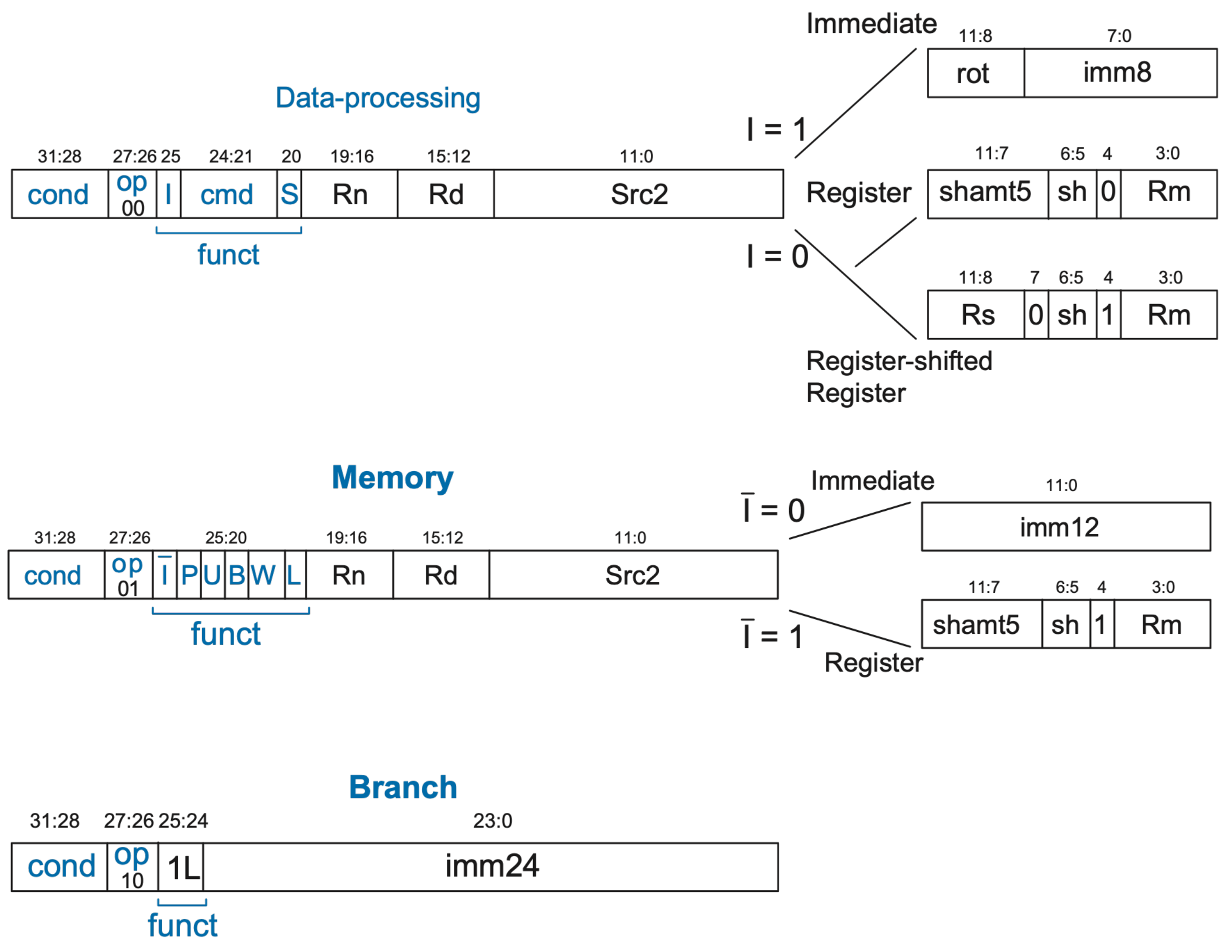
Evolution of ARM Processors

- **Architecture**
- **Microarchitecture**

ARM v4 Architecture

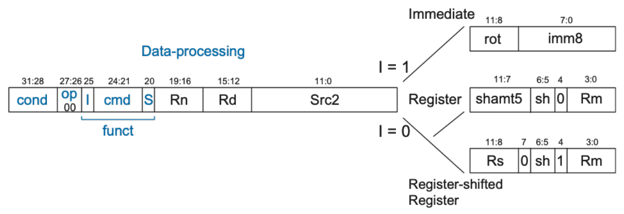
- 16 32-bit registers
  - R15 is the program counter
    - Write to R15 causes a jump
    - Read to R15 returns PC+8
  - R14 is the link register for function calls
- Typical RISC instruction set +
  - Conditional execution
    - Avoids need for some branches
  - Shifter in datapath on second source
    - Helpful for memory addressing
  - Postincrement memory addressing mode

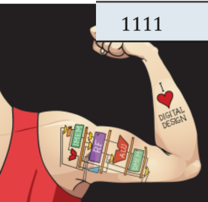
ARM v4 Instruction Formats



Data Processing Instructions

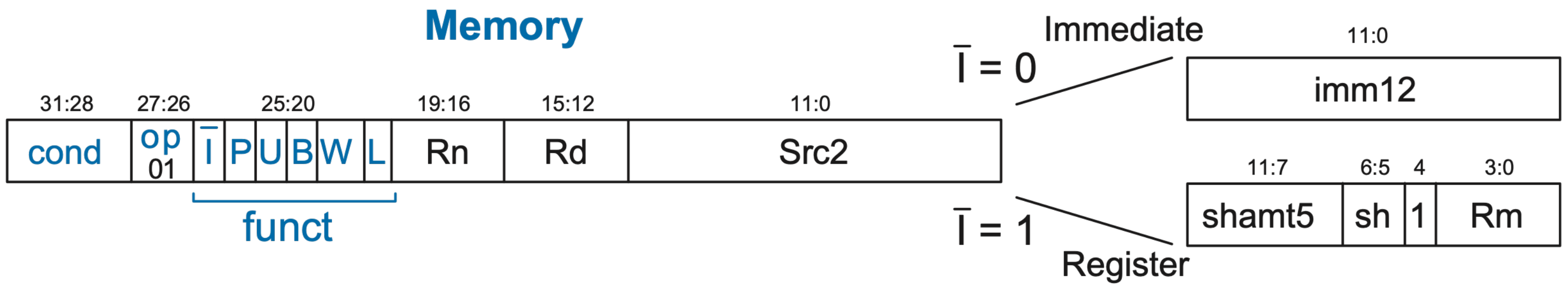
cmd	Name	Description	Operation
0000	AND Rd, Rn, Src2	Bitwise AND	Rd ← Rn & Src2
0001	EOR Rd, Rn, Src2	Bitwise XOR	Rd ← Rn ^ Src2
0010	SUB Rd, Rn, Src2	Subtract	Rd ← Rn - Src2
0011	RSB Rd, Rn, Src2	Reverse Subtract	Rd ← Src2 - Rn
0100	ADD Rd, Rn, Src2	Add	Rd ← Rn+Src2
0101	ADC Rd, Rn, Src2	Add with Carry	Rd ← Rn+Src2+C
0110	SBC Rd, Rn, Src2	Subtract with Carry	Rd ← Rn - Src2 - $\overline{C}$
0111	RSC Rd, Rn, Src2	Reverse Sub w/ Carry	Rd ← Src2 - Rn - $\overline{C}$
1000 (S = 1)	TST Rd, Rn, Src2	Test	Set flags based on Rn & Src2
1001 (S = 1)	TEQ Rd, Rn, Src2	Test Equivalence	Set flags based on Rn ^ Src2
1010 (S = 1)	CMP Rn, Src2	Compare	Set flags based on Rn - Src2
1011 (S = 1)	CMN Rn, Src2	Compare Negative	Set flags based on Rn+Src2
1100	ORR Rd, Rn, Src2	Bitwise OR	Rd ← Rn | Src2
1101	Shifts:		
I = 1 OR (instr$_{11:4}$ = 0)	MOV Rd, Src2	Move	Rd ← Src2
I = 0 AND (sh = 00; instr$_{11:4} \neq$ 0)	LSL Rd, Rm, Rs/shamt5	Logical Shift Left	Rd ← Rm << Src2
I = 0 AND (sh = 01)	LSR Rd, Rm, Rs/shamt5	Logical Shift Right	Rd ← Rm >> Src2
I = 0 AND (sh = 10)	ASR Rd, Rm, Rs/shamt5	Arithmetic Shift Right	Rd ← Rm>>>Src2
I = 0 AND (sh = 11; instr$_{11:7,\ 4}$ = 0)	RRX Rd, Rm, Rs/shamt5	Rotate Right Extend	{Rd, C} ← {C, Rd}
I = 0 AND (sh = 11; instr$_{11:7} \neq$ 0)	ROR Rd, Rm, Rs/shamt5	Rotate Right	Rd ← Rn ror Src2
1110	BIC Rd, Rn, Src2	Bitwise Clear	Rd ← Rn & ~Src2
1111	MVN Rd, Rn, Src2	Bitwise NOT	Rd ← ~Rn





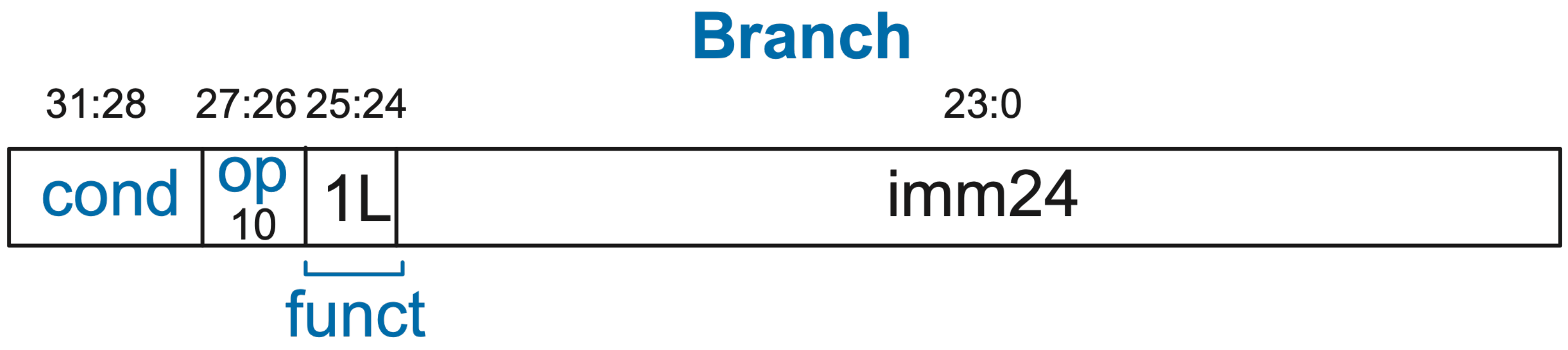
Memory Instructions

op	B	op2	L	Name	Description	Operation
01	0	N/A	0	`STR  Rd, [Rn, ±Src2]`	Store Register	`Mem[Adr] ← Rd`
01	0	N/A	1	`LDR  Rd, [Rn, ±Src2]`	Load Register	`Rd ← Mem[Adr]`
01	1	N/A	0	`STRB Rd, [Rn, ±Src2]`	Store Byte	`Mem[Adr] ←` $Rd_{7:0}$
01	1	N/A	1	`LDRB Rd, [Rn, ±Src2]`	Load Byte	`Rd ←` $Mem[Adr]_{7:0}$



Branch Instructions

L	Name	Description	Operation
0	`B label`	Branch	`PC ← (PC+8)+imm24 << 2`
1	`BL label`	Branch with Link	`LR ← (PC+8) – 4; PC ← (PC+8)+imm24 << 2`

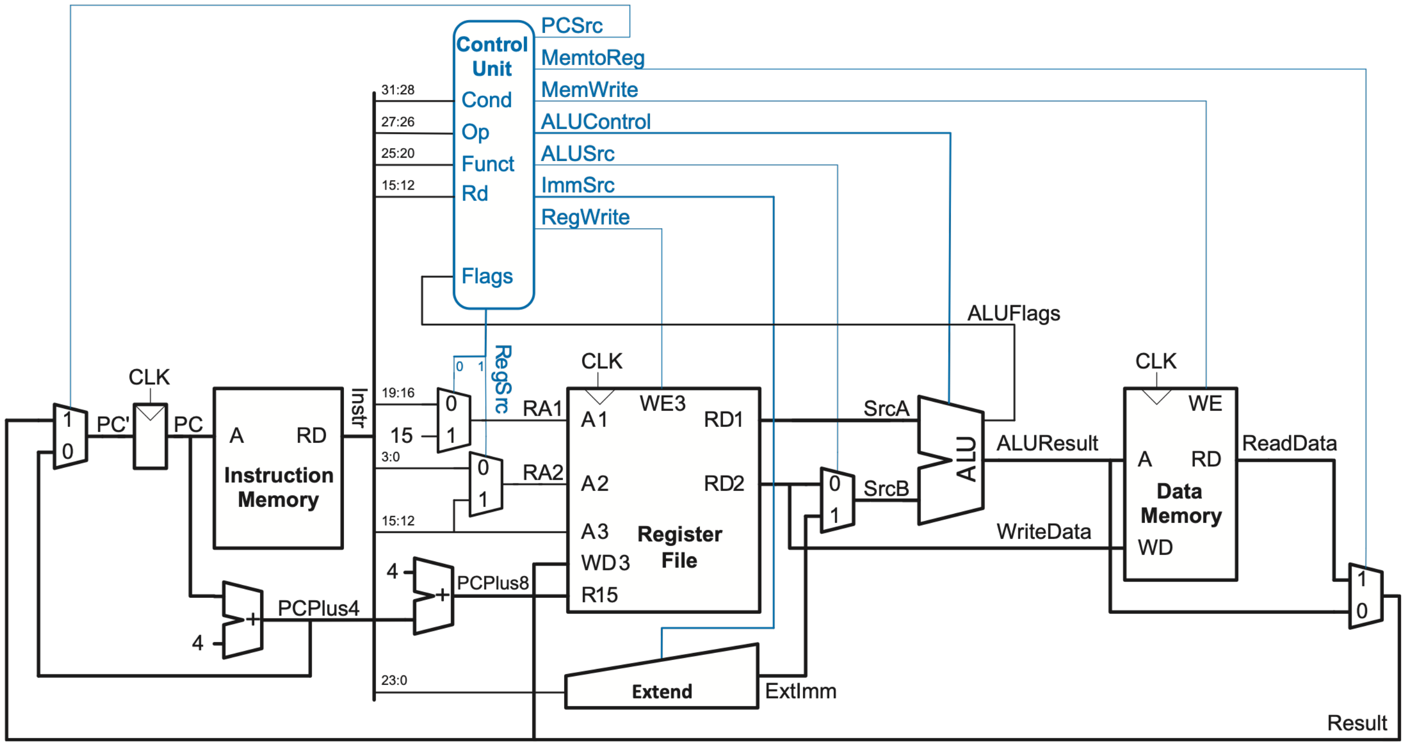


Conditional Execution

cond	Mnemonic	Name	CondEx
0000	EQ	Equal	Z
0001	NE	Not equal	$\overline{Z}$
0010	CS/HS	Carry set / unsigned higher or same	C
0011	CC/LO	Carry clear / unsigned lower	$\overline{C}$
0100	MI	Minus / negative	N
0101	PL	Plus / positive or zero	$\overline{N}$
0110	VS	Overflow / overflow set	V
0111	VC	No overflow / overflow clear	$\overline{V}$
1000	HI	Unsigned higher	$\overline{Z}C$
1001	LS	Unsigned lower or same	Z OR $\overline{C}$
1010	GE	Signed greater than or equal	$\overline{N \oplus V}$
1011	LT	Signed less than	$N \oplus V$
1100	GT	Signed greater than	$\overline{Z}(\overline{N \oplus V})$
1101	LE	Signed less than or equal	Z OR $(N \oplus V)$
1110	AL (or none)	Always / unconditional	Ignored

ELSEVIER

Single Cycle ARM Processor



Similar to RISC-V

PC/R15 input to register file

Source registers fields vary in location

ALU Flags

Architecture Revisions

ARMv2: 1985 Acorn Computer

    26-bit address bus, status in upper PC bits

ARMv3: 32-bit address bus, CPSR

ARMv4: 1993 Halfword loads & stores

    Core ARM instruction set, described here

ARMv4T: Thumb instruction set (16-bit)

ARMv5TE: DSP, Floating point

ARMv6: multimedia, enhanced Thumb

ARMv7: advanced SIMD

ARMv8: 64-bit new instruction set

Thumb

Pack common instructions into 16 bits.

Compete with 16-bit processors for code density.

Limitations:

- R0-R7
- Reuse register as source and dest
- Shorter immediates
- No conditional execution
- Always write status flags

Thumb Example Encodings

15															0	
0	1	0	0	0	0	funct				Rm			Rdn			<funct>S Rdn, Rdn, Rm (data-processing)
0	0	0	ASR	LSR	imm5					Rm			Rd			LSLS / LSRS / ASRS Rd, Rm, #imm5
0	0	0	1	1	1	SUB	imm3			Rm			Rd			ADDS / SUBS Rd, Rm, #imm3
0	0	1	1	SUB	Rdn			imm8								ADDS / SUBS Rdn, Rdn, #imm8
0	1	0	0	0	1	0	0	Rdn [3]	Rm				Rdn[2:0]			ADD Rdn, Rdn, Rm
1	0	1	1	0	0	0	0	SUB	imm7							ADD / SUB SP, SP, #imm7
0	0	1	0	1	Rn			imm8								CMP Rn, #imm8
0	0	1	0	0	Rd			imm8								MOV Rd, #imm8
0	1	0	0	0	1	1	0	Rdn [3]	Rm				Rdn[2:0]			MOV Rdn, Rm
0	1	0	0	0	1	1	1	L	Rm				0	0	0	BX / BLX Rm
1	1	0	1	cond				imm8								B<cond> imm8
1	1	1	0	0	imm8											B imm11
0	1	0	1	L	B	H	Rm			Rn			Rd			STR(B / H) / LDR(B / H) Rd, [Rn, Rm]
0	1	1	0	L	imm5					Rn			Rd			STR / LDR Rd, [Rn, #imm5]
1	0	0	1	L	Rd			imm8								STR / LDR Rd, [SP, #imm8]
0	1	0	0	1	Rd			imm8								LDR Rd, [PC, #imm8]
1	1	1	1	0	imm22[21:11]											
1	1	1	1	1	imm22[10:0]											BL imm22

Digital Signal Processing Instructions

Efficient DSP code, e.g. FFT and FIR

Especially support multiply-accumulate (MAC)

- Z = Z + A * B
- Needs extra source

Often run on shorter data types (e.g. 16 bits)

- Need longer accumulator Z

Saturated arithmetic

DSP Data Types

Type	Sign Bit	Integer Bits	Fractional Bits
short	1	15	0
unsigned short	0	16	0
long	1	31	0
unsigned long	0	32	0
long long	1	63	0
unsigned long long	0	64	0
Q15	1	0	15
Q31	1	0	31

Floating Point

16 64-bit double registers D0-D15

Also usable as 32 32-bit float registers S0-S31

Or 8 128-bit quad registers Q0-Q7 for SIMD

```
VADD.F32 S2, S0, S1
VADD.F64 D2,D0, D1
```

Floating-Point Instructions

Instruction	Function
`VABS  Rd, Rm`	Rd = |Rm|
`VADD  Rd, Rn, Rm`	Rd = Rn + Rm
`VCMP  Rd, Rm`	Compare and set floating-point status flags
`VCVT  Rd, Rm`	Convert between int and float
`VDIV  Rd, Rn, Rm`	Rd = Rn / Rm
`VMLA  Rd, Rn, Rm`	Rd = Rd + Rn * Rm
`VMLS  Rd, Rn, Rm`	Rd = Rd – Rn * Rm
`VMOV  Rd, Rm or #const`	Rd = Rm or constant
`VMUL  Rd, Rn, Rm`	Rd = Rn * Rm
`VNEG  Rd, Rm`	Rd = –Rm
`VNMLA Rd, Rn, Rm`	Rd = –(Rd + Rn * Rm)
`VNMLS Rd, Rn, Rm`	Rd = –(Rd – Rn * Rm)
`VNMUL Rd, Rn, Rm`	Rd = –Rn * Rm
`VSQRT Rd, Rm`	Rd = sqrt(Rm)
`VSUB  Rd, Rn, Rm`	Rd = Rn – Rm

SIMD Instructions

Packed operations on 8-64 bit data

in 64-128 bit registers D, Q

.I8, .I16, .I32, .I64

.F32, .F64

```
VADD.I8 D2, D1, D0
VADD.I32 Q2, Q1, Q0
VADD.F32 D2, D1, D0
```

64-bit Instruction Set

32-bit addresses access 2^{32} = 4GB of RAM

too little for PCs or phone nowadays

64-bit addresses access 2^{64} bytes (huge)

ARMv8 uses 64-bit registers and addresses

Instructions still 32 bits

ARMv8 Registers

Expands to 32 64-bit registers

X0-X30

PC and SP are no longer general purpose regs

X30 acts as the link regiser

No X31, but Zero Reg (ZR) instead

Hardwired to 0

ARMv8 Instruction Encodings

Different instruction encoding in 64-bit mode

5 bit register addresses

Remove conditionals from all but branch instrs.

Expands SIMD registers

Cryptography instructions

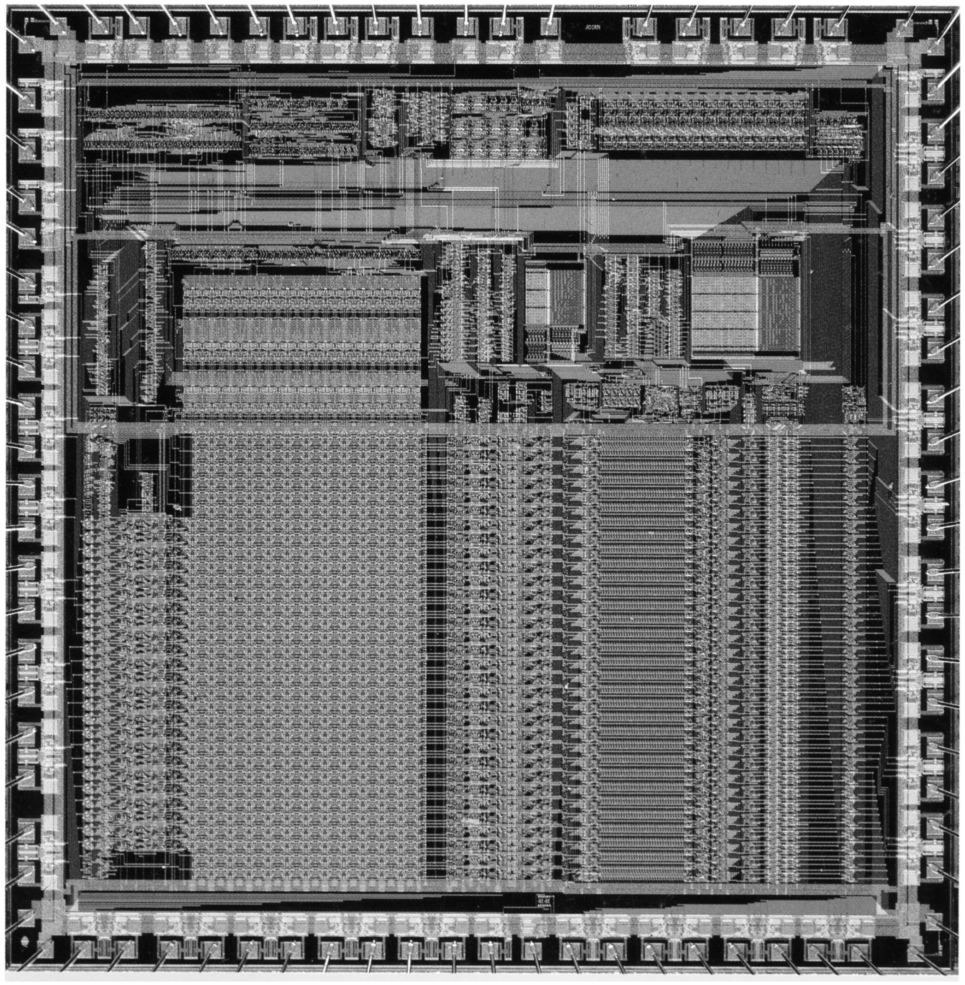
Boot in 64-bit mode

Can change to 32-bit for compatibility

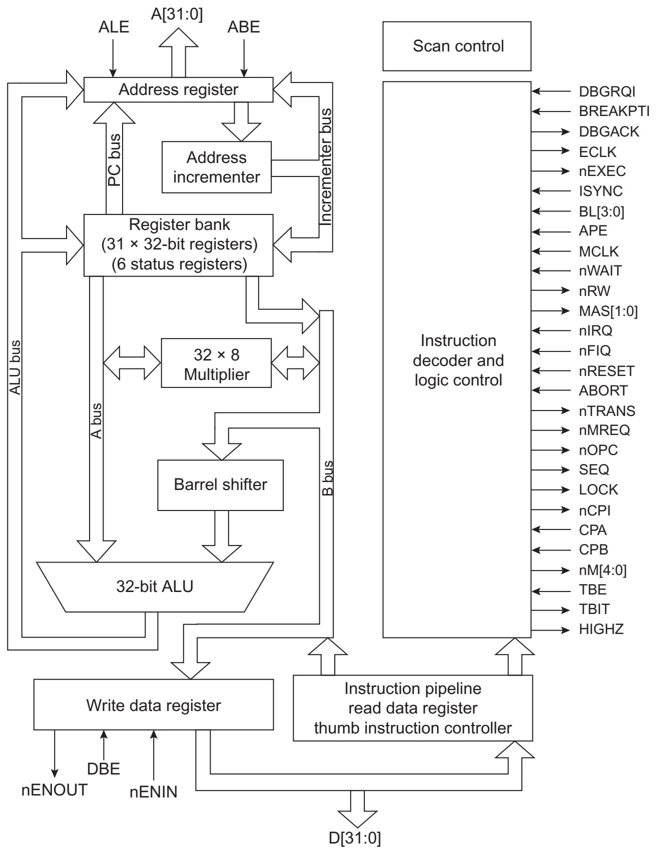
Evolution of ARM Processors

Microarchitecture	Year	Architecture	Pipeline Depth	DMIPS/ MHz	Representative Frequency (MHz)	L1 Cache	Relative Size
ARM1	1985	v1	3	0.33	8	N/A	0.1
ARM6	1992	v3	3	0.65	30	4 KB unified	0.6
ARM7	1994	v4T	3	0.9	100	0–8 KB unified	1
ARM9E	1999	v5TE	5	1.1	300	0–16 KB I + D	3
ARM11	2002	v6	8	1.25	700	4–64 KB I + D	30
Cortex-A9	2009	v7	8	2.5	1000	16–64 KB I + D	100
Cortex-A7	2011	v7	8	1.9	1500	8–64 KB I + D	40
Cortex-A15	2011	v7	15	3.5	2000	32 KB I + D	240
Cortex-M0 +	2012	v7M	2	0.93	60–250	None	0.3
Cortex-A53	2012	v8	8	2.3	1500	8–64 KB I + D	50
Cortex-A57	2012	v8	15	4.1	2000	48 KB I + 32 KB D	300

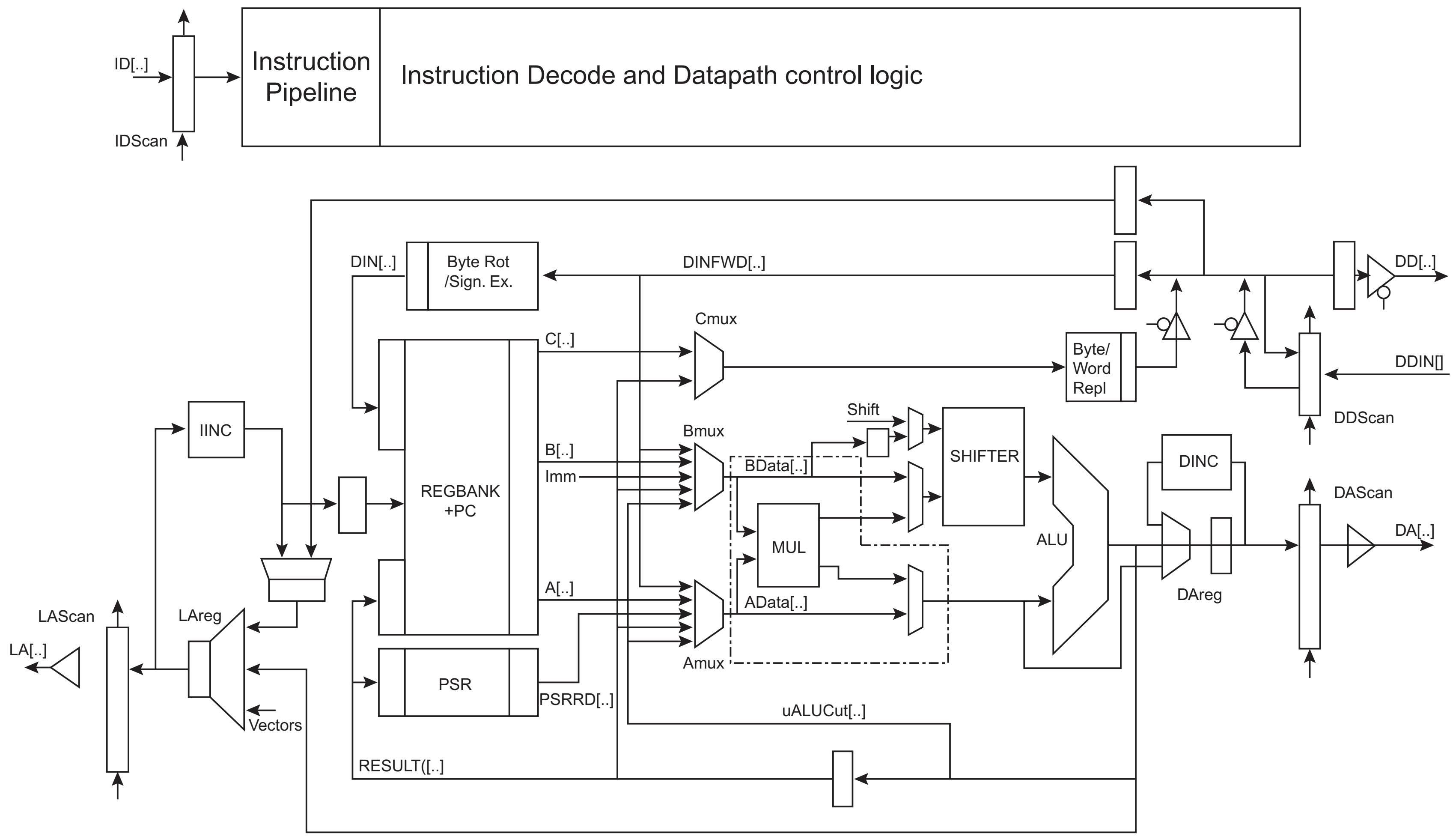
ARM1 die photo



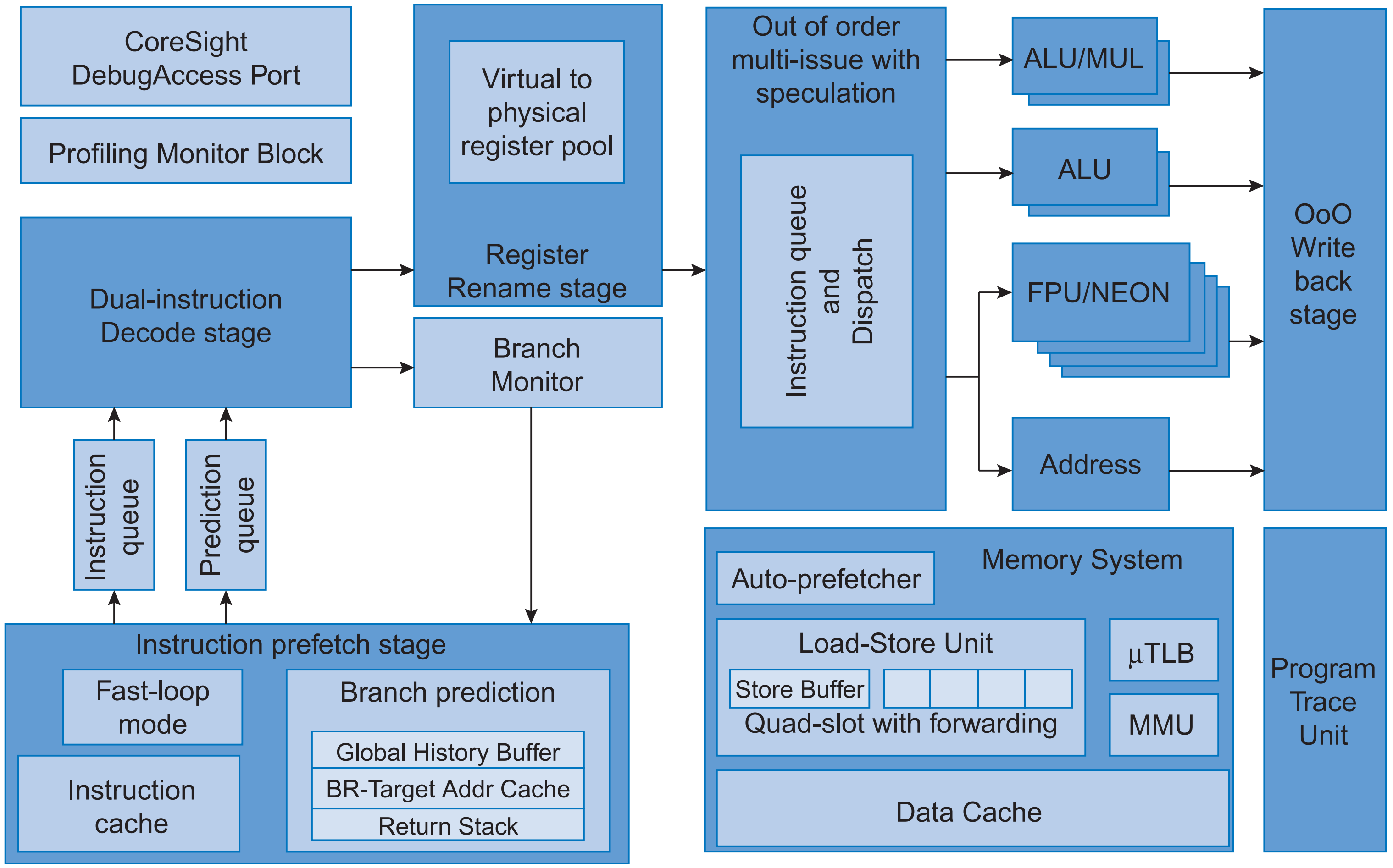
ARM7 Block Diagram (v4T)



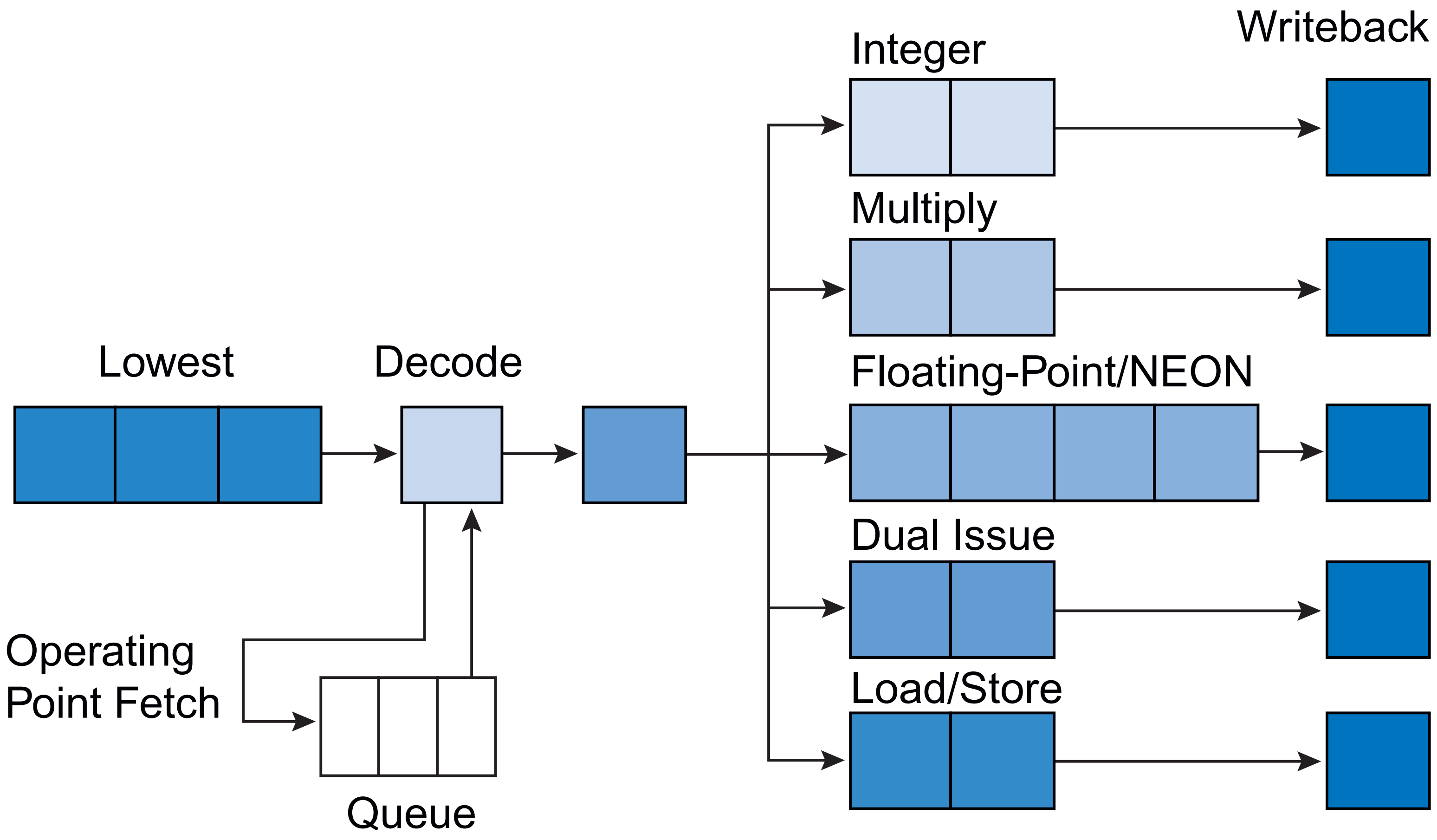
ARM9 Block Diagram (v5TE)



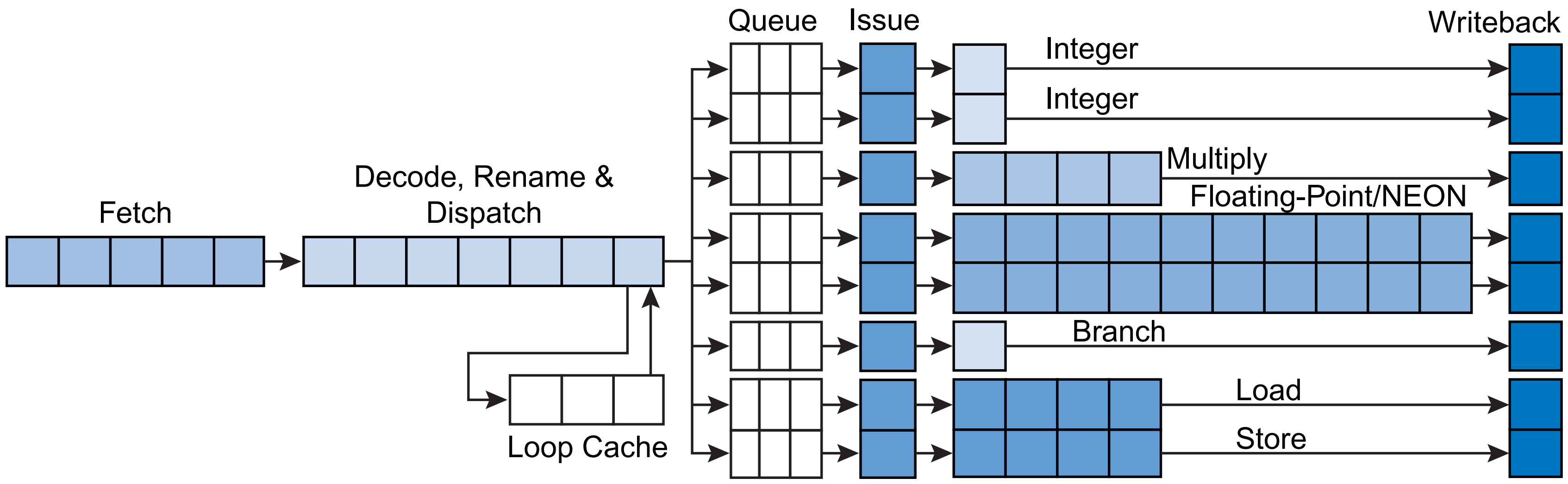
Cortex A9 Block Diagram



Cortex A7 Pipeline



Cortex A15 Pipeline



Recent Developments

64-bit ARMv8

Bit.LITTLE / DynamIQ

Cortex A73/A53

Cortex A75/A55

Cortex A76

Cortex A77

2019 Benchmarks

Manufacturer	Chip	Product	Process	Big Core	Frequency (GHz)	SpecInt2006 Energy (kJ)	SpecInt2006 Speed	Core Size (mm^2)
Apple	A12	iPhone XS/XR	TSMC 7nm	Vortex	2.49	9.5	45.3	2.07
Samsung	Exynos 9820	Galaxy S10	Samsung 8nm	M4	2.73	14.3	26.3	
Qualcomm	Snapdragon 855	Galaxy S10+	TSMC 7 nm	A76	2.84	9.7	26.7	
Huawei	Kirin 980	Huawei P30	TSMC 7nm	A76	2.6	9.5	25.7	
Apple	A11	iPhone 8/X	TSCM 10 nm	Monsoon	2.39	10.6	36.8	2.68
Samsung	Exynos 9810	Galaxy S9	Samsung 10 nm	M3	2.70	20.1	23.8	3.46
Qualcomm	Snapdragon 845	Note S9	Samsung 10 nm	A75	2.80	12.5	17.7	
Qualcomm	Snapdragon 835	Galaxy S8	10 nm	A73	2.45	13.3	13.6	

2019 Benchmarks

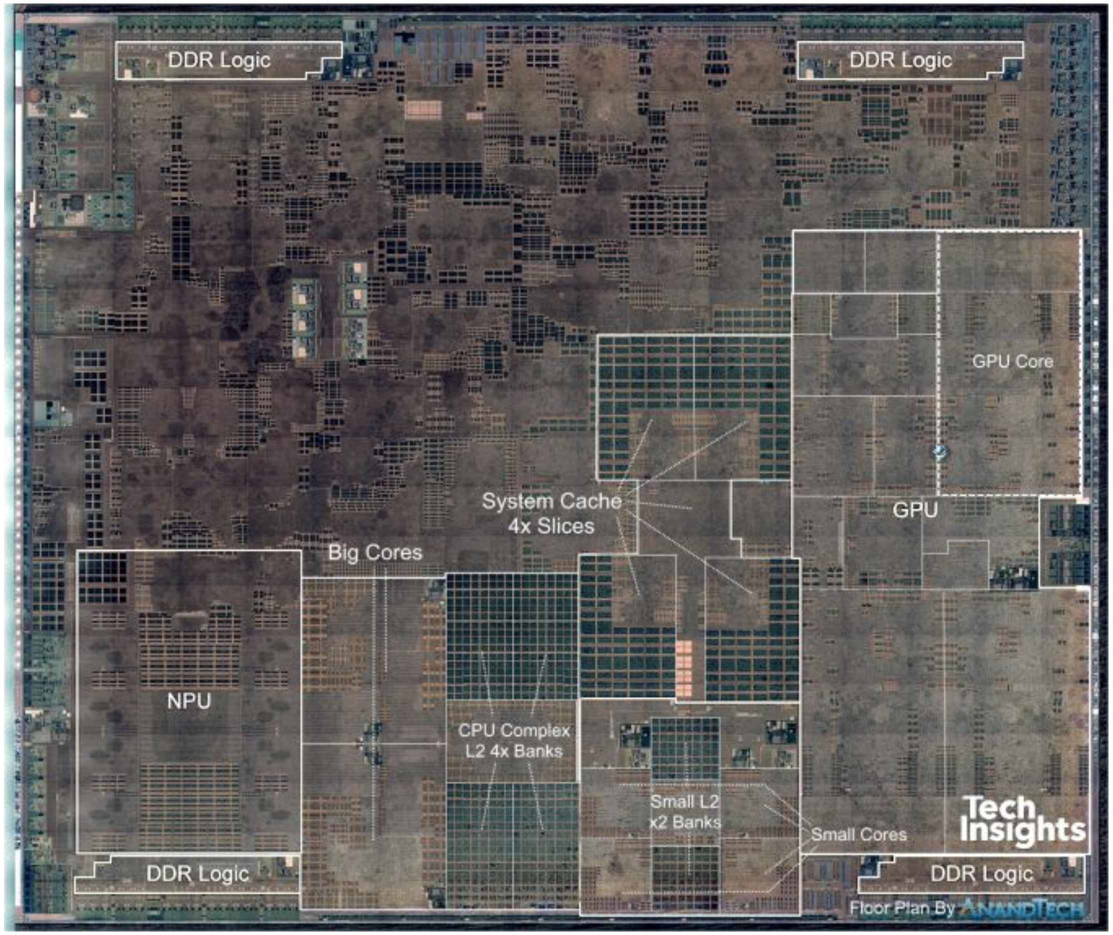
Manufacturer	Chip	Cores	Cache Size	Die Size (mm^2)	Transistors (Billion)
Apple	A12	2 Vortex, 4 Tempest	8 MB	83	6.9B
Samsung	Exynos 9820	2 M4, 2 A75, 4 A55	2 MB	127	
Qualcomm	Snapdragon 855	4 A76, 4 A55	5 MB	73	
Huawei	Kirin 980	4 A76, 4 A55	4 MB	74	6.9
Apple	A11	2 Monsoon, 4 Mistral	8 MB	88	4.3
Samsung	Exynos 9810	4 M3	4 MB	119	
Qualcomm	Snapdragon 845	4 A75, 4 A55	2 MB	95	5.3
Qualcomm	Snapdragon 835	4 A73, 4 A53	2 MB	72	3

Microarchitectures

Manufacturer	Core	Decode	Out-of-Order	Pipeline Stages
Apple	Vortex (A12) Monsoon (A11)	7-Way	Yes	
Samsung	M4	6-Way	Yes	16
ARM	Cortex A76	4-Way	Yes	11-13
ARM	Cortex A75	3-Way	Yes	11-13
ARM	Cortex A55	2-Way	No	8

Apple A12

2018
TSMC 7 nm
83 mm^2
2x Vortex
    2.49 GHz
    512 KB L2$
4x Tempest
    1.8 GHz
    128 KB L2$
4 MB L3$
Mali GPU
LTE Modem



Samsung Exynos 9820

2018

TSMC 7 nm

74 mm^2

2x M4 2.6 GHz

512 KB L2$

2x A75 1.92 GHz

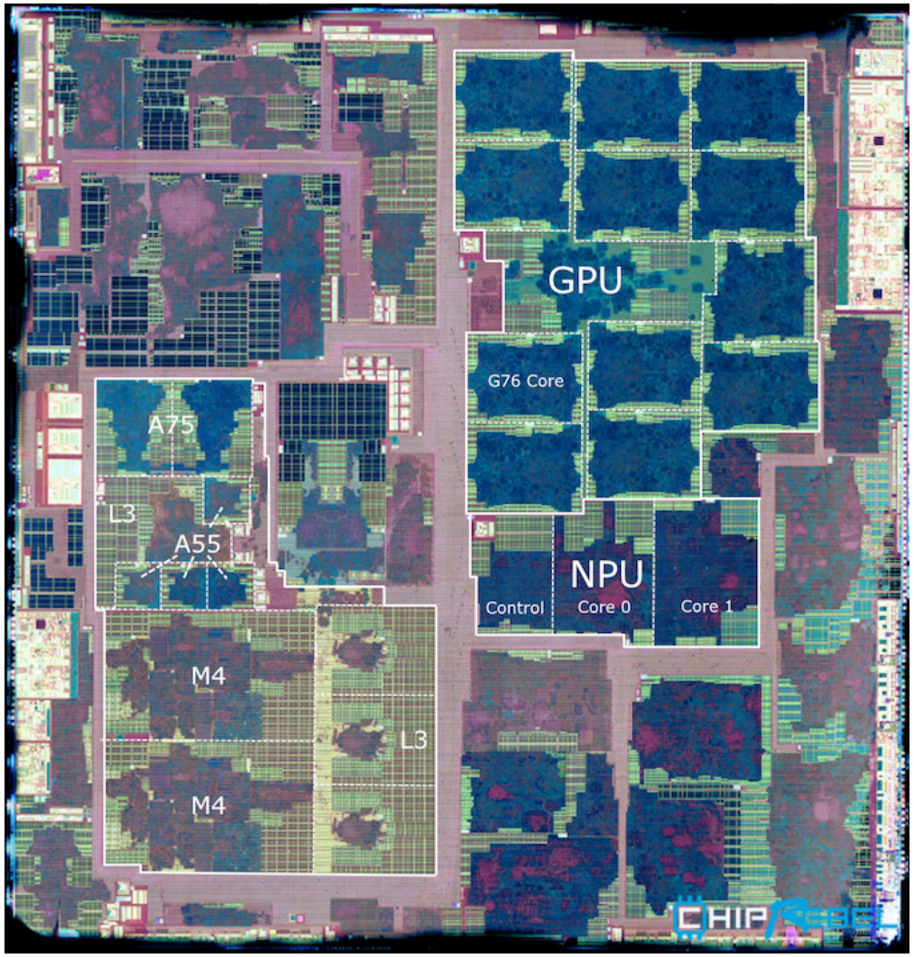
4x A55 1.8 GHz

128 KB L2$

4 MB L3$

Mali GPU

LTE Modem

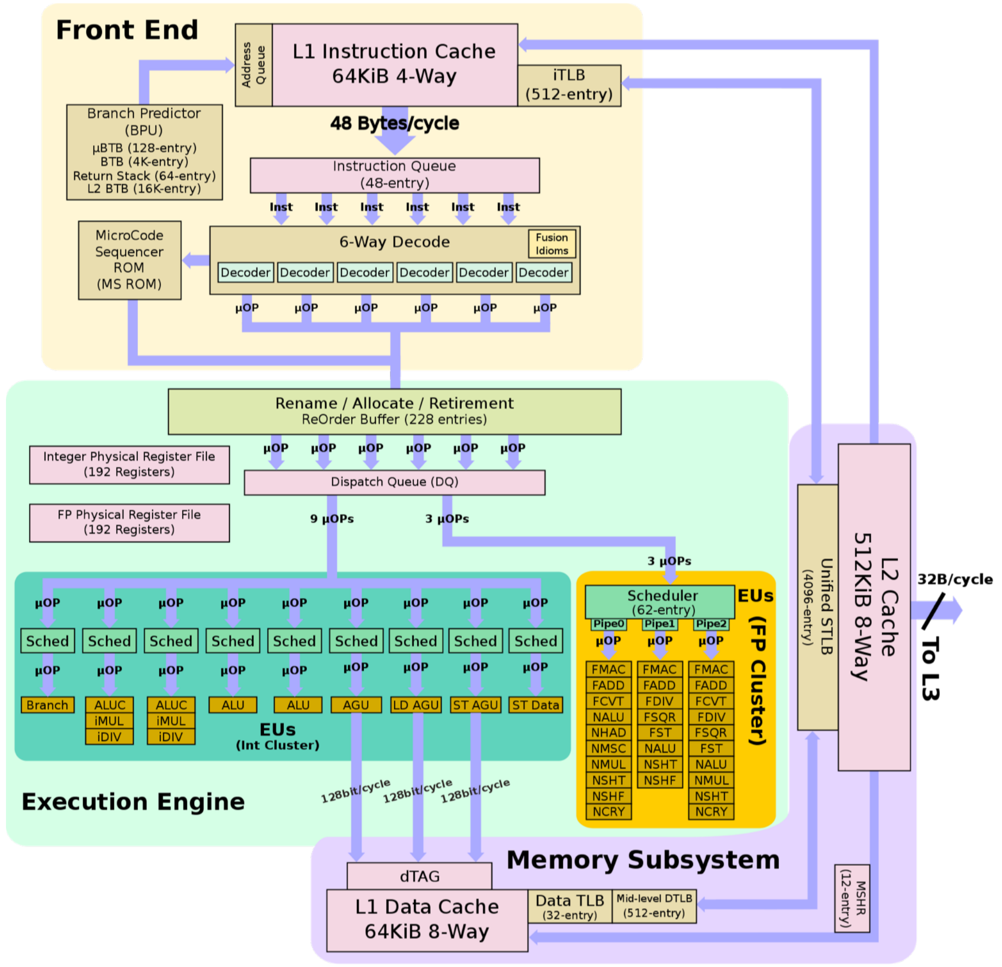


Samsung Exynos 9820 M4 uArch

6-way decode

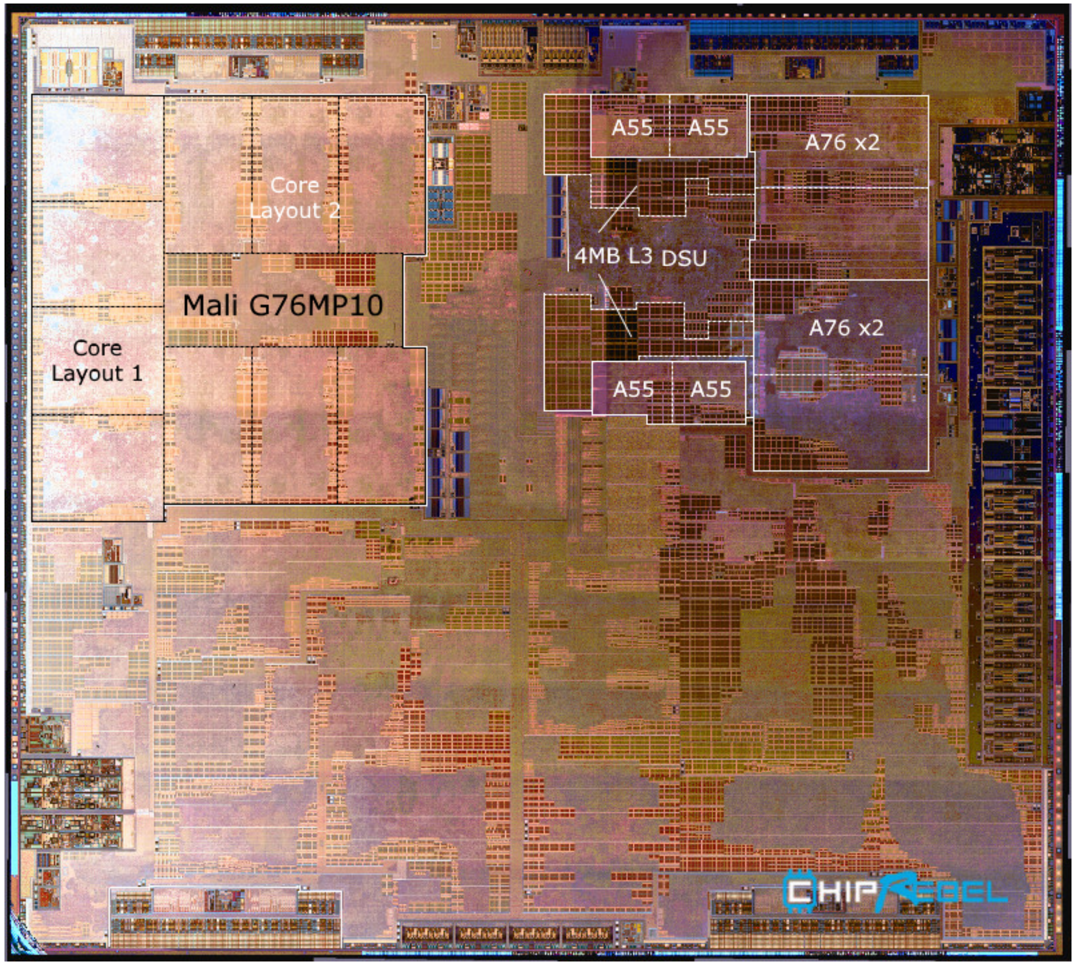
12 execution pipes

- 1 branch
- 2 alu
- 2 complex
- 4 ld/store
- 3 FP pipes


Front End
Address Queue
L1 Instruction Cache 64KiB 4-Way
iTLB (512-entry)
Branch Predictor (BPU) µBTB (128-entry) BTB (4K-entry) Return Stack (64-entry) L2 BTB (16K-entry)
48 Bytes/cycle
Instruction Queue (48-entry)
Inst
MicroCode Sequencer ROM (MS ROM)
6-Way Decode
Fusion Idioms
Decoder
µOP
Rename / Allocate / Retirement
ReOrder Buffer (228 entries)
Integer Physical Register File (192 Registers)
FP Physical Register File (192 Registers)
Dispatch Queue (DQ)
9 µOPs
3 µOPs
Sched
Branch
ALUC iMUL iDIV
ALU
AGU
LD AGU
ST AGU
ST Data
EUs (Int Cluster)
Scheduler (62-entry)
Pipe0 Pipe1 Pipe2
FMAC FADD FCVT NALU NHAD NMSC NMUL NSHT NSHF NCRY
FMAC FADD FDIV FSQR FST NALU NSHT NSHF
FMAC FADD FCVT FDIV FSQR FST NALU NMUL NSHT NCRY
EUs (FP Cluster)
Execution Engine
128bit/cycle
Unified STLB (4096-entry)
L2 Cache 512KiB 8-Way
32B/cycle
To L3
Memory Subsystem
dTAG
L1 Data Cache 64KiB 8-Way
Data TLB (32-entry)
Mid-level DTLB (512-entry)
MSHR (12-entry)

Huawei Kirin 980

2018
TSMC 7 nm
74 mm^2
2x A76 2.6 GHz
512 KB L2$
2x A76 1.92 GHz
4x A55 1.8 GHz
128 KB L2$
4 MB L3$
Mali GPU
LTE Modem

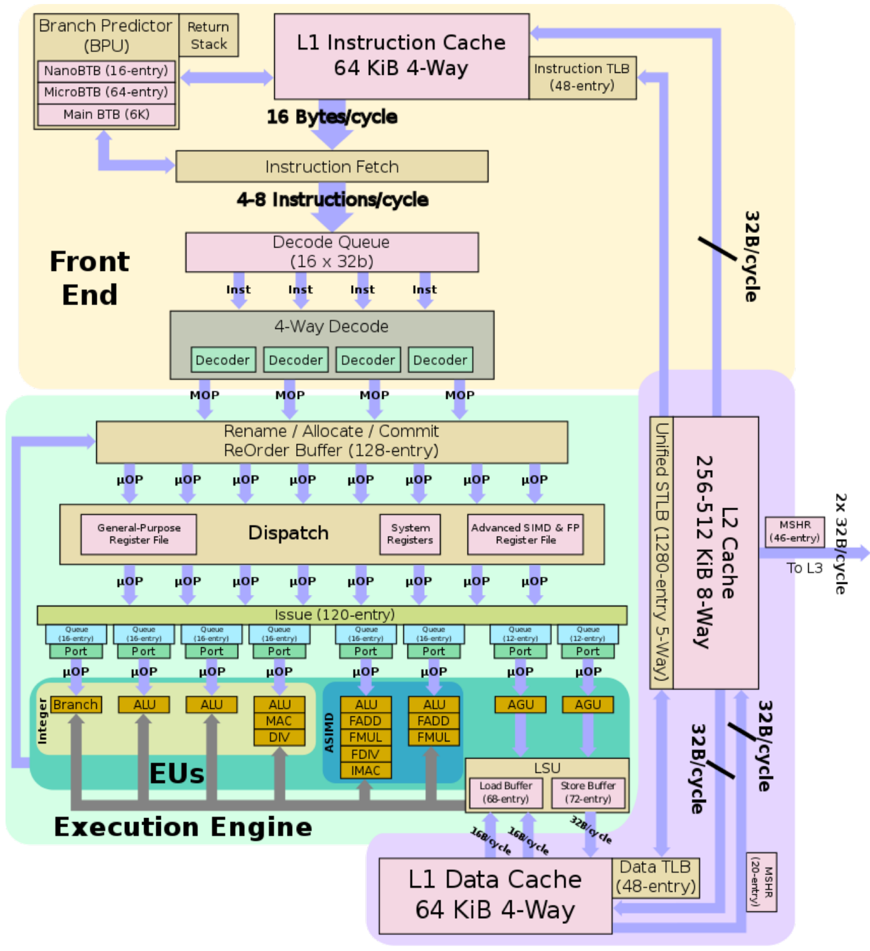


ARM A76 uArch

4-way decode

8 execution pipes

- 1 branch
- 2 alu
- 1 complex
- 2 ld/store
- 2 FP pipes

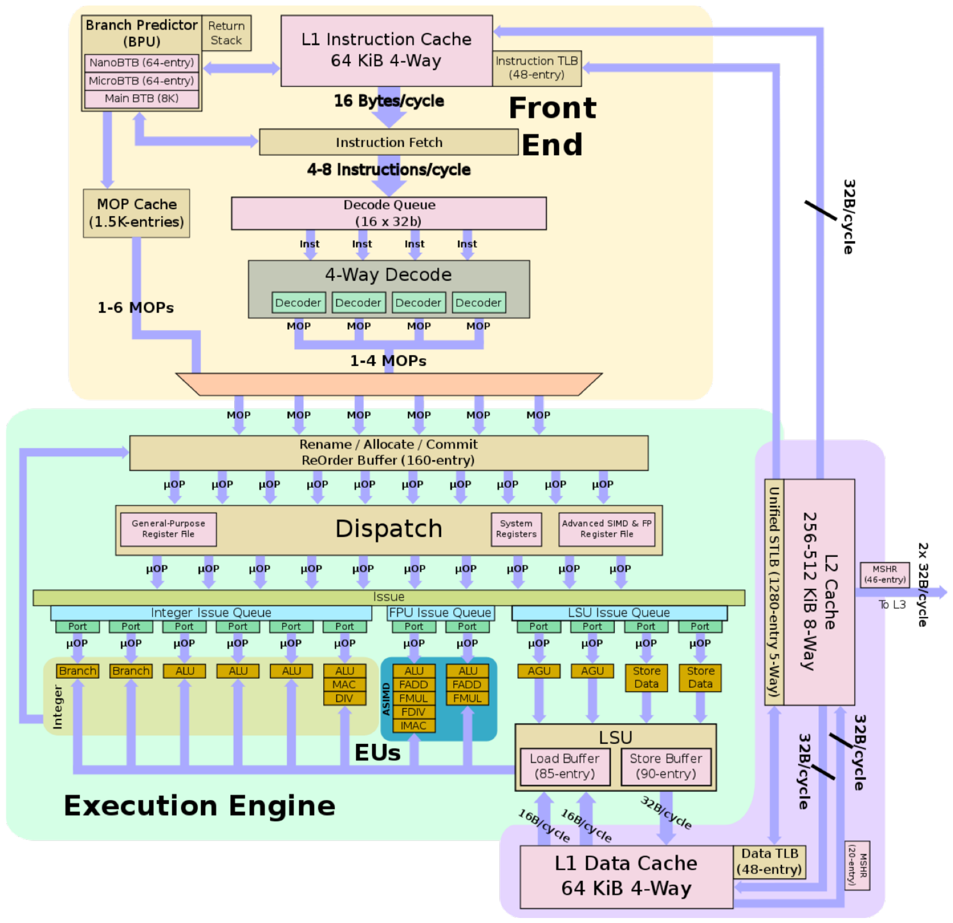


ARM A77 uArch

4-way decode

12 execution pipes

- 2 branch
- 3 alu
- 1 complex
- 4 ld/store
- 2 FP pipes



Branch Predictor (BPU)
Return Stack
NanoBTB (64-entry)
MicroBTB (64-entry)
Main BTB (8K)
L1 Instruction Cache 64 KiB 4-Way
Instruction TLB (48-entry)
16 Bytes/cycle
Front End
Instruction Fetch
4-8 Instructions/cycle
MOP Cache (1.5K-entries)
Decode Queue (16 x 32b)
Inst
4-Way Decode
Decoder
MOP
1-6 MOPs
1-4 MOPs
Rename / Allocate / Commit
ReOrder Buffer (160-entry)
µOP
General-Purpose Register File
Dispatch
System Registers
Advanced SIMD & FP Register File
Issue
Integer Issue Queue
FPU Issue Queue
LSU Issue Queue
Port
Branch
ALU
ALU MAC DIV
ALU FADD FMUL FDIV IMAC
ALU FADD FMUL
AGU
Store Data
Integer
ASIMD
EUs
LSU
Load Buffer (85-entry)
Store Buffer (90-entry)
Execution Engine
16B/cycle
32B/cycle
L1 Data Cache 64 KiB 4-Way
Data TLB (48-entry)
MSHR (20-entry)
Unified STLB (1280-entry 5-Way)
L2 Cache 256-512 KiB 8-Way
MSHR (46-entry)
To L3
2x 32B/cycle