18-447
ARM ISA
Recitation #1

Prof. Onur Mutlu

Carnegie Mellon University

Spring 2014, 1/14/2013

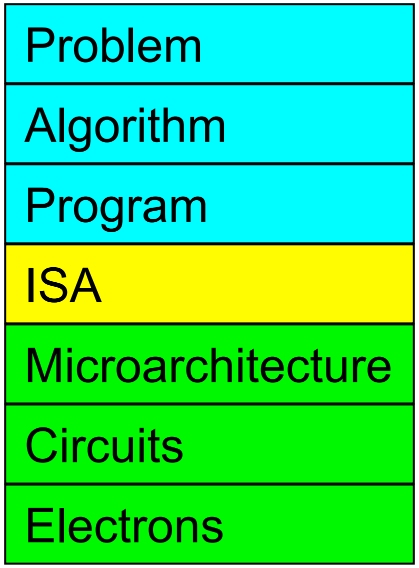
What is Computer Architecture?

- **ISA+implementation definition:** The science and art of designing, selecting, and interconnecting hardware components and designing the hardware/software interface to create a computing system that meets functional, performance, energy consumption, cost, and other specific goals.

- **Traditional (only ISA) definition:** “The term *architecture* is used here to describe the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behavior as distinct from the organization of the dataflow and controls, the logic design, and the physical implementation.” *Gene Amdahl*, IBM Journal of R&D, April 1964

ISA vs. Microarchitecture

- ISA
  - Agreed upon interface between software and hardware
    - SW/compiler assumes, HW promises
  - What the software writer needs to know to write and debug system/user programs
- Microarchitecture
  - Specific implementation of an ISA
  - Not visible to the software
- Microprocessor
  - **ISA, uarch**, circuits
  - "Architecture" = ISA + microarchitecture



Problem

Algorithm

Program

ISA

Microarchitecture

Circuits

Electrons

ISA

- Instructions
  - Opcodes, Addressing Modes, Data Types
  - Instruction Types and Formats
  - Registers, Condition Codes
- Memory
  - Address space, Addressability, Alignment
  - Virtual memory management
- Call, Interrupt/Exception Handling
- Access Control, Priority/Privilege
- I/O: memory-mapped vs. instr.
- Task/thread Management
- Power and Thermal Management
- Multi-threading support, Multiprocessor support

Intel® 64 and IA-32 Architectures
Software Developer's Manual

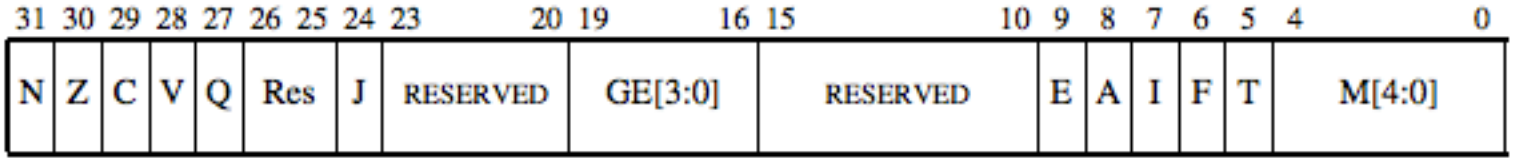
Volume 1:
Basic Architecture

ARM ISA – What is it?

- ARM is a RISC architecture
- Used for mobile computing
- Features
  - Load/Store Architecture
  - Conditional Execution
  - Inline barrel shifter
  - Multiple execution modes involving banked registers
  - Many different addressing modes
  - Thumb mode (16-bit mode)

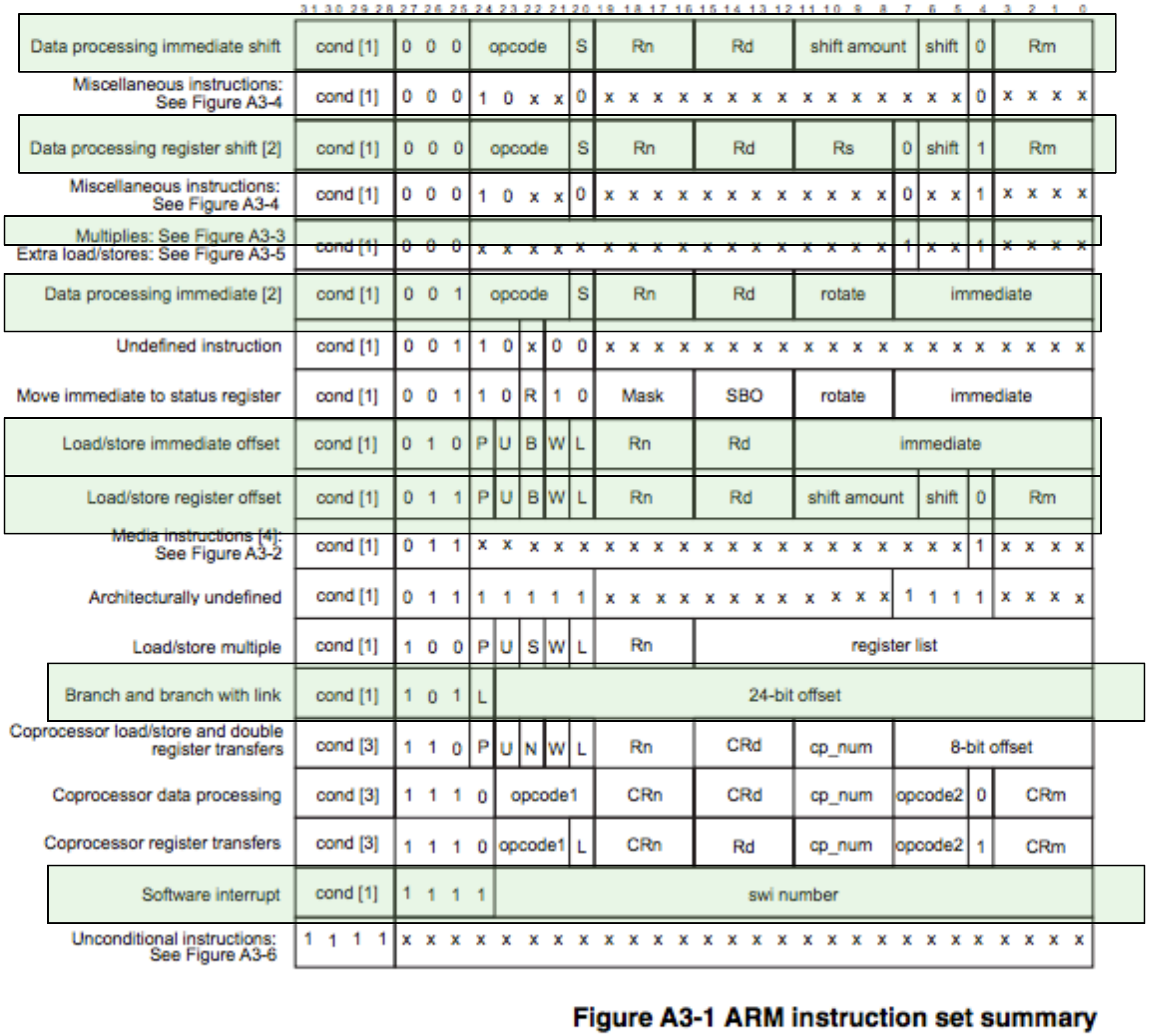
ARM Basics - Registers

- 16 General Purpose Registers
  - R15 is the PC
  - R14 is the linker addr
  - R13 is the stack pointer
- CPSR and SPSP



31	30	29	28	27	26 25	24	23 20	19 16	15 10	9	8	7	6	5	4 0
N	Z	C	V	Q	Res	J	RESERVED	GE[3:0]	RESERVED	E	A	I	F	T	M[4:0]

ARM Basics – Instruction Encodings

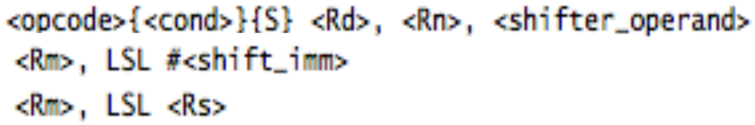


	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data processing immediate shift	cond [1]				0	0	0	opcode				S	Rn				Rd				shift amount					shift		0	Rm			
Miscellaneous instructions: See Figure A3-4	cond [1]				0	0	0	1	0	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	x	x	x	x
Data processing register shift [2]	cond [1]				0	0	0	opcode				S	Rn				Rd				Rs				0	shift		1	Rm			
Miscellaneous instructions: See Figure A3-4	cond [1]				0	0	0	1	0	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	0	x	x	1	x	x	x	x
Multiplies: See Figure A3-3 Extra load/stores: See Figure A3-5	cond [1]				0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	1	x	x	x	x
Data processing immediate [2]	cond [1]				0	0	1	opcode				S	Rn				Rd				rotate				immediate							
Undefined instruction	cond [1]				0	0	1	1	0	x	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Move immediate to status register	cond [1]				0	0	1	1	0	R	1	0	Mask				SBO				rotate				immediate							
Load/store immediate offset	cond [1]				0	1	0	P	U	B	W	L	Rn				Rd				immediate											
Load/store register offset	cond [1]				0	1	1	P	U	B	W	L	Rn				Rd				shift amount					shift		0	Rm			
Media instructions [4]: See Figure A3-2	cond [1]				0	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x
Architecturally undefined	cond [1]				0	1	1	1	1	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x	1	1	1	1	x	x	x	x
Load/store multiple	cond [1]				1	0	0	P	U	S	W	L	Rn				register list															
Branch and branch with link	cond [1]				1	0	1	L	24-bit offset																							
Coprocessor load/store and double register transfers	cond [3]				1	1	0	P	U	N	W	L	Rn				CRd				cp_num				8-bit offset							
Coprocessor data processing	cond [3]				1	1	1	0	opcode1				CRn				CRd				cp_num				opcode2			0	CRm			
Coprocessor register transfers	cond [3]				1	1	1	0	opcode1			L	CRn				Rd				cp_num				opcode2			1	CRm			
Software interrupt	cond [1]				1	1	1	1	swi number																							
Unconditional instructions: See Figure A3-6	1	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure A3-1 ARM instruction set summary

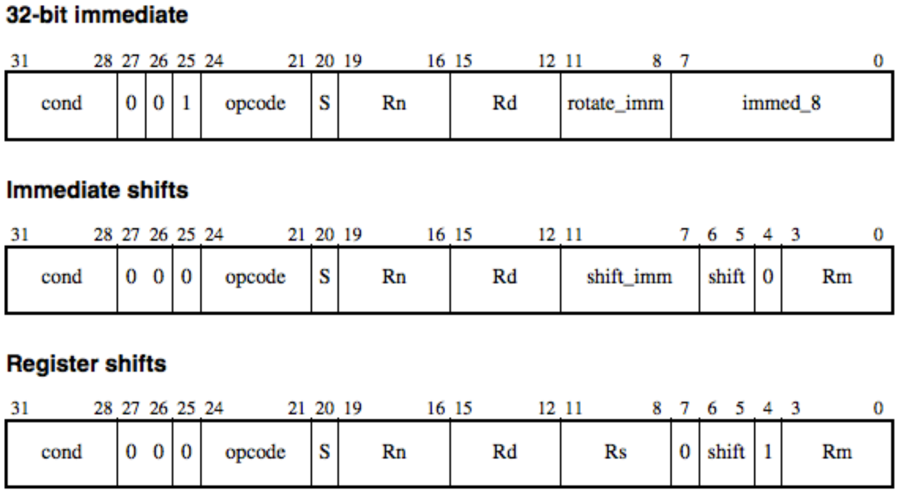
ARM ISA - Barrel Shifter

- 4 Different types of shifts: LSL, LSR, ASR, ROR (special case RRX)
- Assembler code:



```
<opcode>{<cond>}{S} <Rd>, <Rn>, <shifter_operand>
 <Rm>, LSL #<shift_imm>
 <Rm>, LSL <Rs>
```

- Encoding:



32-bit immediate

31 28	27	26	25	24 21	20	19 16	15 12	11 8	7 0
cond	0	0	1	opcode	S	Rn	Rd	rotate_imm	immed_8

Immediate shifts

31 28	27	26	25	24 21	20	19 16	15 12	11 7	6 5	4	3 0
cond	0	0	0	opcode	S	Rn	Rd	shift_imm	shift	0	Rm

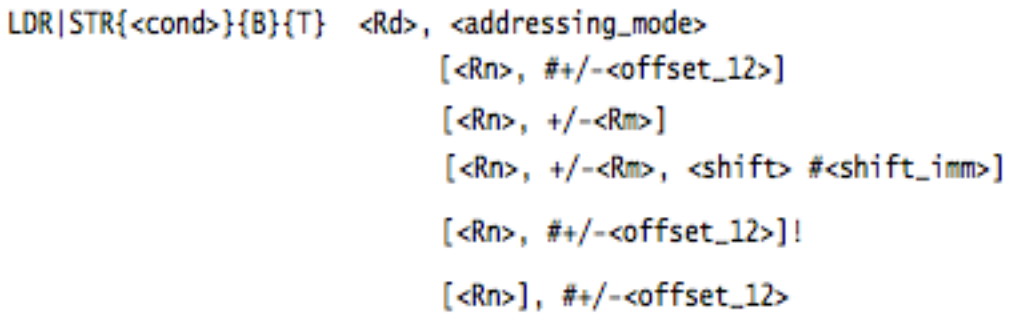
Register shifts

31 28	27	26	25	24 21	20	19 16	15 12	11 8	7	6 5	4	3 0
cond	0	0	0	opcode	S	Rn	Rd	Rs	0	shift	1	Rm

More info on page A5-3 of *ARM Architecture Reference Manual*

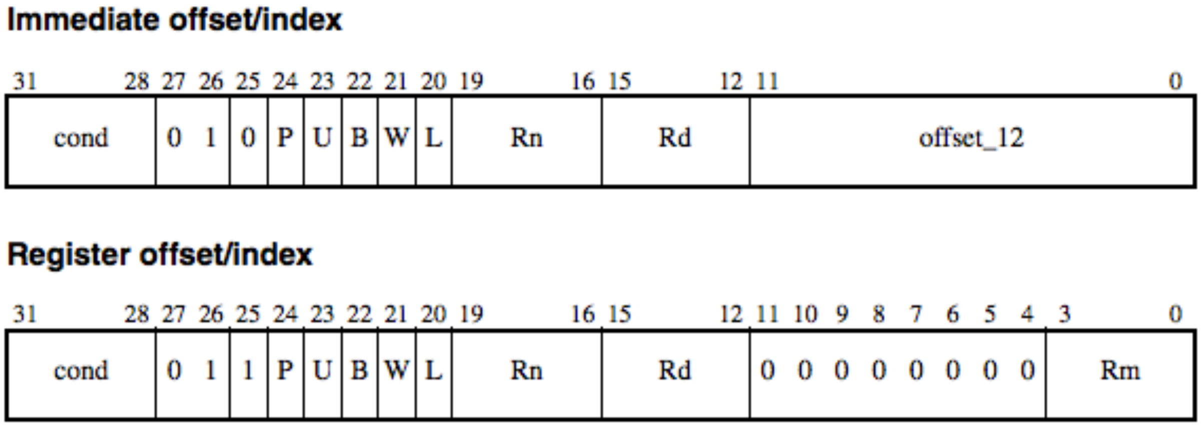
ARM ISA - Addressing Mode

- Many different modes (about 9)
  - Only worry about 2 for this class
- Assembler Code:



```
LDR|STR{<cond>}{B}{T}  <Rd>, <addressing_mode>
                             [<Rn>, #+/-<offset_12>]
                             [<Rn>, +/-<Rm>]
                             [<Rn>, +/-<Rm>, <shift> #<shift_imm>]
                             [<Rn>, #+/-<offset_12>]!
                             [<Rn>], #+/-<offset_12>
```

- Encoding:



Immediate offset/index

31 28	27	26	25	24	23	22	21	20	19 16	15 12	11 0
cond	0	1	0	P	U	B	W	L	Rn	Rd	offset_12

Register offset/index

31 28	27	26	25	24	23	22	21	20	19 16	15 12	11	10	9	8	7	6	5	4	3 0
cond	0	1	1	P	U	B	W	L	Rn	Rd	0	0	0	0	0	0	0	0	Rm

More info on page A5-19 of *ARM Architecture Reference Manual*