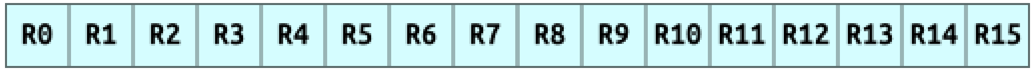
Instructions

CS2400

Spring 2020

Review

ARM has sixteen registers visible at any one time. They are named R0 to R15. All are 32 bits wide.



R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15

All of the registers are general purpose, save for:

- R13 / SP
  - which holds the *stack pointer*.
- R14 / LR
  - the link register which holds the caller's return address.
- R15 / PC
  - which holds the *program counter*.

In addition to the main registers there is also a status register:

CPSR is the *current program status register*. This holds flags: results of arithmetic and logical operations.

Instructions

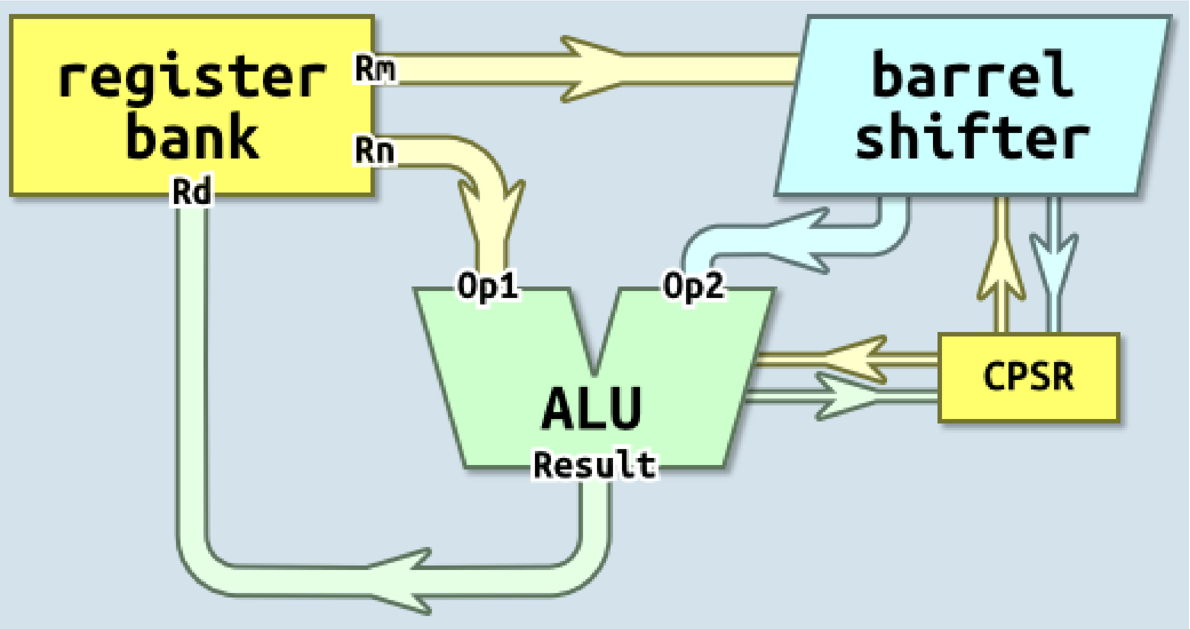
- Instructions - Language of computers
- Vocabulary – **Instruction Set**

- Different computers have different instruction sets

Instructions

- Syntax
- <operation>{cond}{flags} Rd,Rn,Operand2
    - <operation>
        - A three-letter mnemonic, e.g. MOV or ADD.
    - {cond}
        - An optional two-letter condition code, e.g. EQ or CS.
    - {flags}
        - An optional additional flags. e.g. S.
    - Rd
        - The destination register.
    - Rn
        - The first source register.
    - Operand2
        - A flexible second operand.

Organisation



Movement

- **<operation>{cond}{S} Rd,Operand2**
- Examples

  - MOV r0, #42
    - Move the constant 42 into register R0.
  - MOV r2, r3
    - Move the contents of register R3 into register R2.
  - MVN r1, r0
    - R1 = NOT(R0) = -43
  - MOV r0, r0
    - A NOP (no operation) instruction.

Arithmetic Operations

- Example
- ADD a, b, c ; The sum of b and c is placed in a
- ADD a, a, d ; The sum of b, c, and d is now in a
- ADD a, a, e ; The sum of b, c, d, and e is now in a

LEGv8 operands

- 32 registers - X0...X30, XZR
- 2^{62} memory doublewords - Memory[0], Memory [4], ...,

  ..Memory[4,611,686,018,427,387,904]

Instructions

Category	Instruction	Example	Meaning	Comments
Arithmeti c	add	ADD X1, X2, X3	X1 = X2 + X3	Three register operands
	subtract	SUB X1, X2, X3	X1 = X2 - X3	Three register operands
	add immediate	ADDI X1, X2, 20	X1 = X2 + 20	Used to add constants
	subtract immediate	SUBI X1, X2, 20	X1 = X2 - 20	Used to subtract constants
	add and set flags	ADDS X1, X2, X3	X1 = X2 + X3	Add, set condition codes
	subtract and set flags	SUBS X1, X2, X3	X1 = X2 - X3	Subtract, set condition codes
	add immediate and set flags	ADDIS X1, X2, 20	X1 = X2 + 20	Add constant, set condition codes
	subtract immediate and set flags	SUBIS X1, X2, 20	X1 = X2 - 20	Subtract constant, set condition codes

Operands

- The reason for the limit of 32 registers may be found in the second of our three underlying design principles of hardware technology:
  - Smallest is faster

- A very large number of registers may increase the clock cycle time simply because it takes electronic signals longer when they must travel farther.

- In LEGv8 architecture Register is 64 bit wide.

Example

- C code:
- `f = (g + h) - (i + j);`
- Instructions
-
  ```
  add t0, g, h     ; temp t0 = g + h
  add t1, i, j     ; temp t1 = i + j
  sub f, t0, t1    ; f = t0 – t1
  ```

Label

- Labels are alphanumeric names used to define the starting location of a block of statements.

- ARM assembler has reserved first character of a line for the label field and it should be left blank for the instructions with no labels.

- Example :
- stop B

Assembler Directives

- Assembler directives are commands to the assembler that direct the assembly process. Assembler directives are also called pseudo opcodes or pseudo-operations.
- Some tasks performed by these directives are:
- 1. Assign the program to certain areas in memory.
- 2. Define symbols.
- 3. Designate areas of memory for data storage.
- 4. Place tables or other fixed data in memory.
- 5. Allow references to other programs.

- Example : AREA,CODE,DATA,READONLY ……,DCD,DCB ….etc….

Load –Store architecture

- ARM is a *load-store architecture language*:
  - You must load values into registers in order to operate upon them.
  - No instructions directly operate on values in memory.

  - Word -4 Bytes

Memory Operands

- LDR R4,=A
- LDR R0,[R4]
- LDR R4,=X
- STR R3,[R4]
-
- A DCD 0x45
-
-                 AREA Data1,DATA,READWRITE
- X DCD 0

Example Program

```
       LDR R4,=A
      LDR R0,[R4]
      LDR R4,=C
      LDR R1,[R4]
      ADD R3,R0,R1
      LDR R4,=D
      LDR R2,[R4]
      SUB R3,R3,R2
      LDR R4,=X
      STR R3,[R4]
st        B  st
A  DCD  0x45
C  DCD  0x25
D  DCD  0x05
             AREA Data1,DATA,READWRITE
X  DCD 0
             END
```

Data Definition Directive -DCD

- The DCD (Define Constant Data) directive allocates one or more words of memory, aligned on 4-byte boundaries, and defines the initial runtime contents of the memory. & is a synonym for DCD.

- The syntax of DCD is:

- {label} DCD expression{,expression}

DCB Directive

- The DCB directive allocates one or more bytes of memory, and defines the initial runtime contents of the memory.

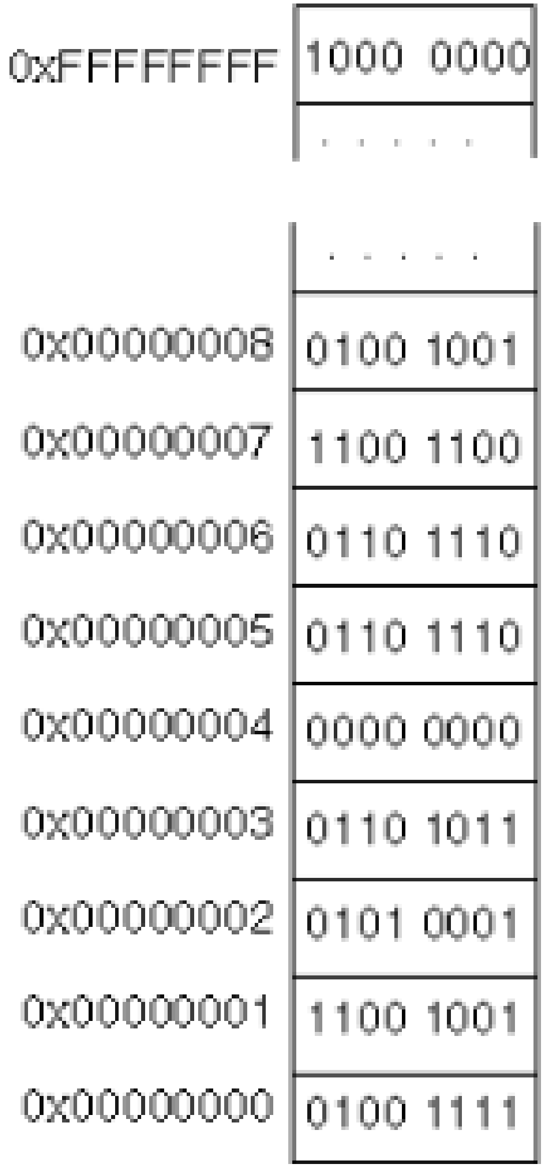
- num DCB 0x48
- C_string   DCB "C_string",0

Memory Operands

- ***Data transfer instruction***: A command that moves data between memory and registers.

- *Address* : A value used to delineate the location of a specific data element within a memory array.

- The data transfer instruction that copies data from memory to a register is traditionally called ***load***.



0xFFFFFFFF	1000 0000
	. . . . .
	. . . . .
0x00000008	0100 1001
0x00000007	1100 1100
0x00000006	0110 1110
0x00000005	0110 1110
0x00000004	0000 0000
0x00000003	0110 1011
0x00000002	0101 0001
0x00000001	1100 1001
0x00000000	0100 1111

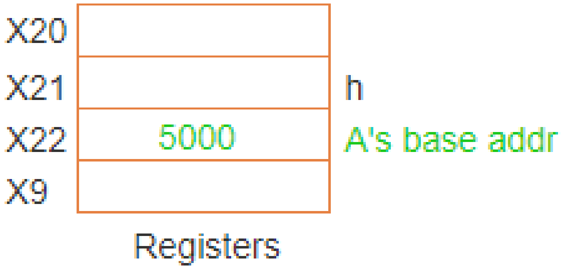
Memory Array

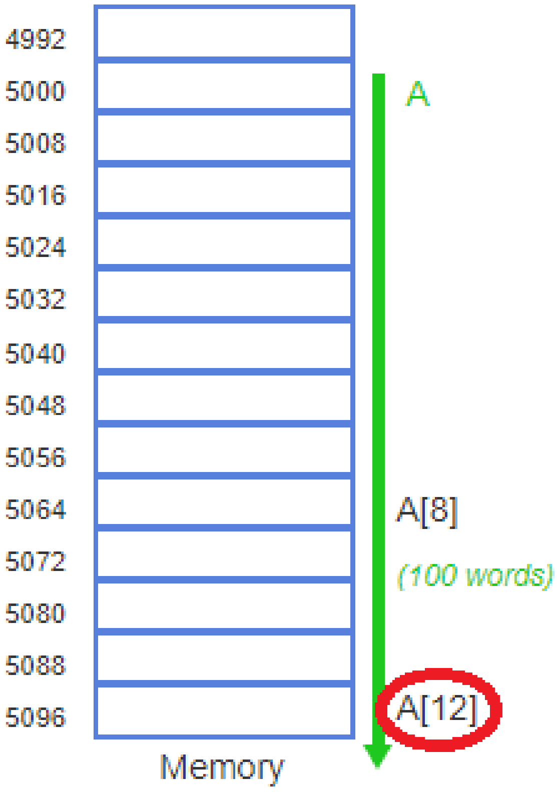
- a ***base address*** is the starting address of an array in memory
- a ***base register*** is a register that holds an array's base address
- ***offset*** is a constant value added to a base address to locate a particular array element .

- LDR X9, [X22, #8]

Memory Address

A[12] = h + A[8];





```
 LDR  X9, [X22, #64] // Temporary reg X9 gets A[8]
ADD   X9, X21, X9    // Temporary reg X9 gets h + A[8]

 STR  X9, [X22, #96] // Stores h + A[8] back into A[12]
```

Q

- If X22 has 1000, X9 has 77, and memory locations 1000, 1008, and 1016 have 10, 15, 20 respectively, what do those locations have after the following instruction?

- STR X9, [X22, #8]

- 10 , 77 ,20

Load and Store

- LDR -> Load word to register
- STR ->Save word from register
- LDRB -> Load Byte to register
- STRB ->Save Byte from register
- LDRH ->Load Halfword to register
- STRH ->Save Halfword from register

Q

- Write an ARM assembly program to accept 2 numbers from variable A and B .Find their sum and store it in variable C.

LEGv8 Instructions

- *R-type* Instructions

Opcode	Rm	shamt	Rn	Rd
11 bits	5 bits	6 bits	5 bits	5 bits

LEGv8 Instructions

- *D-type* Instructions

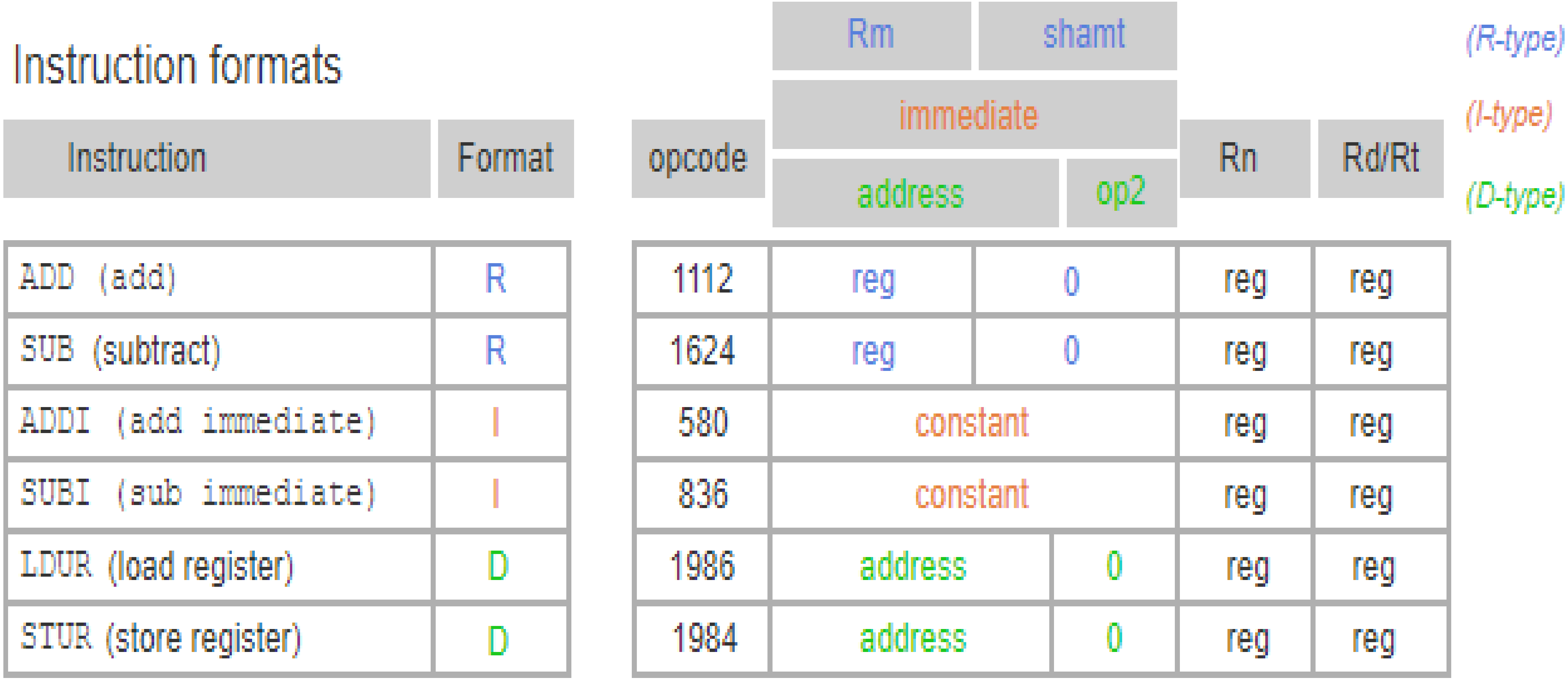
Opcode	address	op 2	Rn	Rt
11 bits	9 bits	2 bits	5 bits	5 bits

LEGv8 Instructions

- *I-type* format

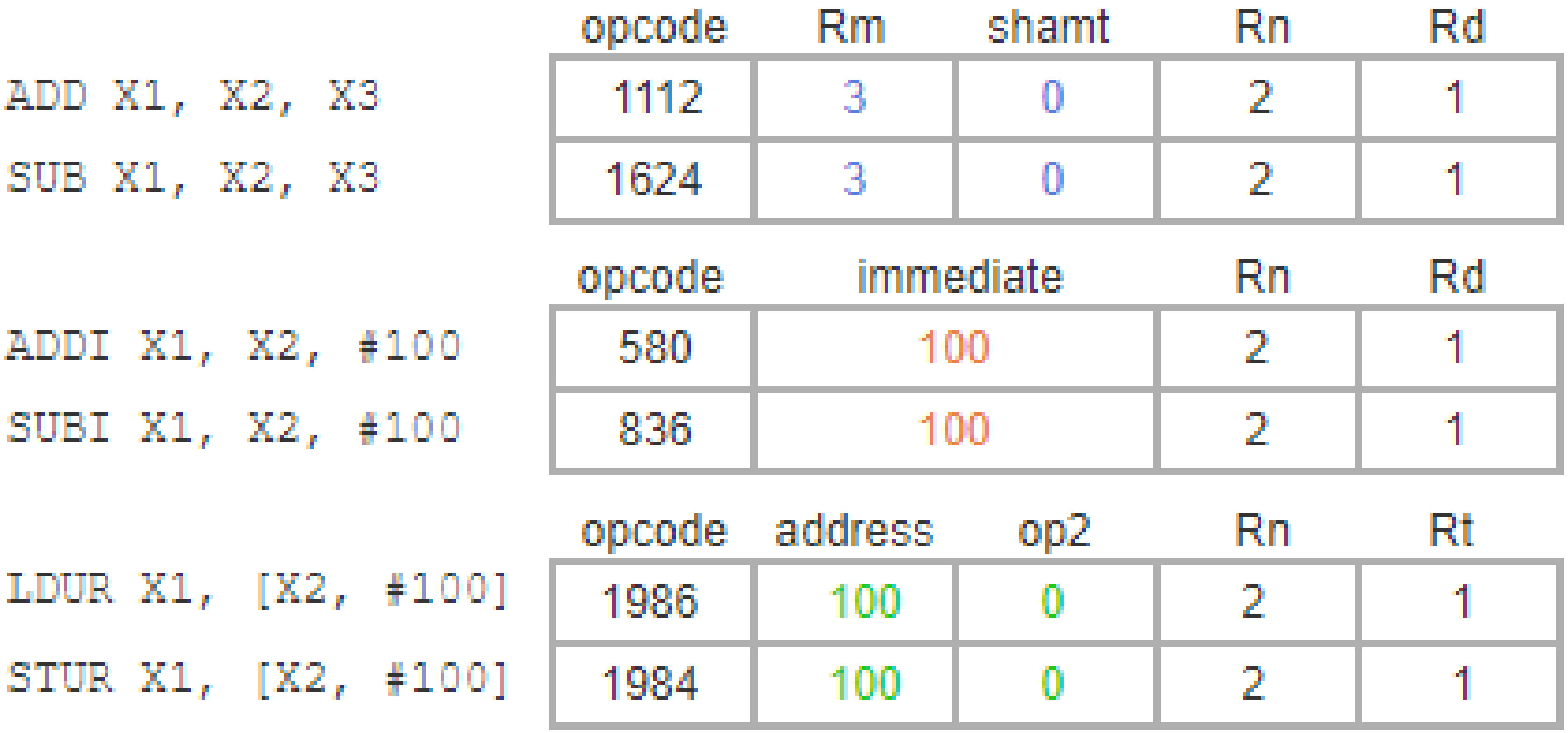
Opcode	immediate	Rn	Rd
10 bits	12 bits	5 bits	5 bits

Instruction formats



			Rm	shamt			(R-type)
			immediate				(I-type)
Instruction	Format	opcode	address	op2	Rn	Rd/Rt	(D-type)
ADD (add)	R	1112	reg	0	reg	reg	
SUB (subtract)	R	1624	reg	0	reg	reg	
ADDI (add immediate)	I	580	constant		reg	reg	
SUBI (sub immediate)	I	836	constant		reg	reg	
LDUR (load register)	D	1986	address	0	reg	reg	
STUR (store register)	D	1984	address	0	reg	reg	

Sample Instructions



	opcode	Rm	shamt	Rn	Rd
`ADD X1, X2, X3`	1112	3	0	2	1
`SUB X1, X2, X3`	1624	3	0	2	1

	opcode	immediate	Rn	Rd
`ADDI X1, X2, #100`	580	100	2	1
`SUBI X1, X2, #100`	836	100	2	1

	opcode	address	op2	Rn	Rt
`LDUR X1, [X2, #100]`	1986	100	0	2	1
`STUR X1, [X2, #100]`	1984	100	0	2	1

Programming Question

- Write an ARM assembly program to find the value for the given equation with decimal values of x=20 and y=90 .Store the value in z. Use only ADD operation.

- z= x-y

Next Class

- Logical operation
- Multiplication
- Division