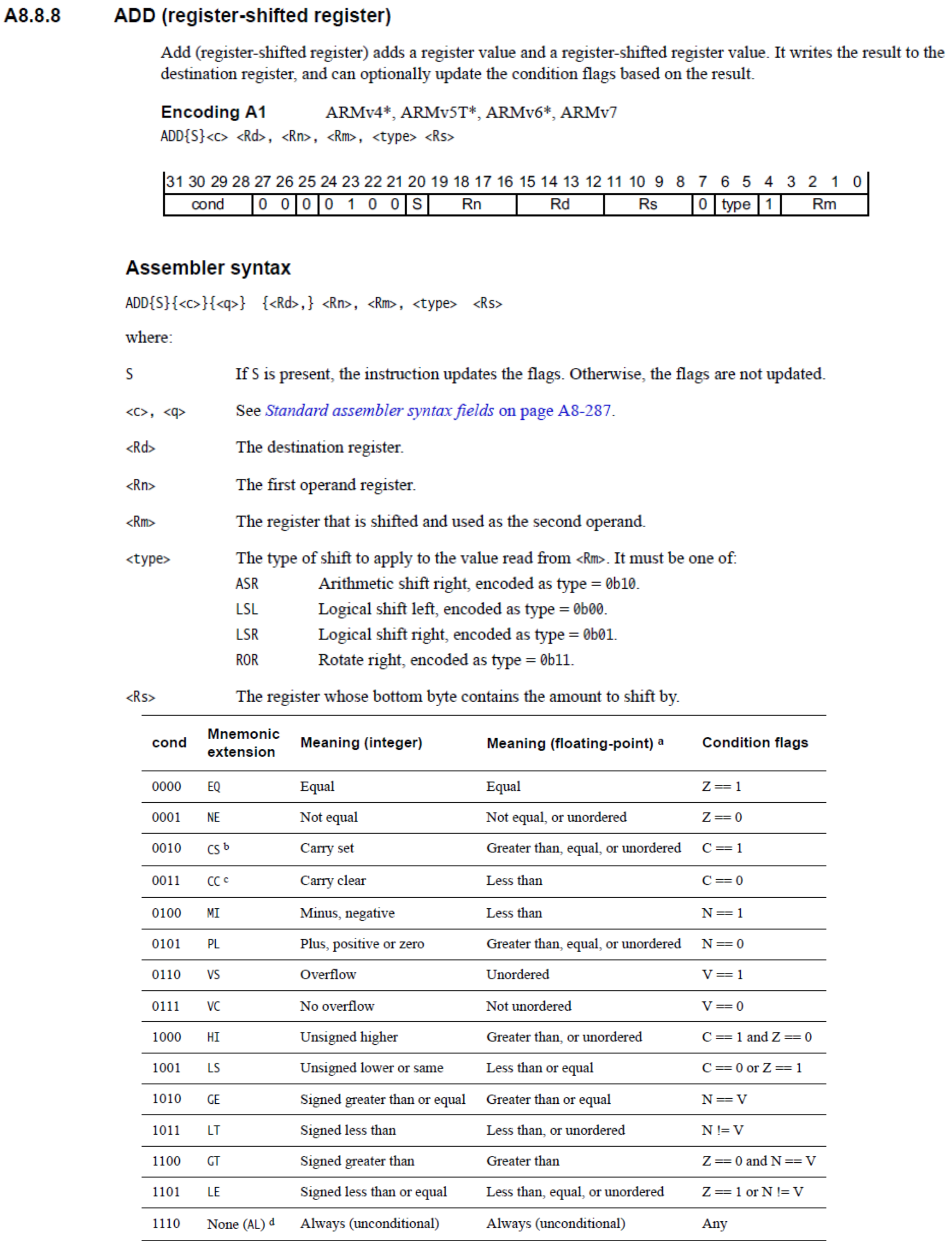
Microprocessors Problem Set #2

Revision: February 4, 2026

RealDigital
www.realdigital.org

1. (8 points) Show the 32-bit opcode for the instruction ADDNE R0, R1, R2, LSR R3 in the table below.

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



A8.8.8 ADD (register-shifted register)

Add (register-shifted register) adds a register value and a register-shifted register value. It writes the result to the destination register, and can optionally update the condition flags based on the result.

Encoding A1 ARMv4*, ARMv5T*, ARMv6*, ARMv7

`ADD{S}<c> <Rd>, <Rn>, <Rm>, <type> <Rs>`

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
cond				0	0	0	0	1	0	0	S	Rn				Rd				Rs				0	type		1	Rm			

Assembler syntax

`ADD{S}{<c>}{<q>} {<Rd>,} <Rn>, <Rm>, <type> <Rs>`

where:

S If S is present, the instruction updates the flags. Otherwise, the flags are not updated.

<c>, <q> See *Standard assembler syntax fields* on page A8-287.

<Rd> The destination register.

<Rn> The first operand register.

<Rm> The register that is shifted and used as the second operand.

<type> The type of shift to apply to the value read from <Rm>. It must be one of:

ASR	Arithmetic shift right, encoded as type = 0b10.
LSL	Logical shift left, encoded as type = 0b00.
LSR	Logical shift right, encoded as type = 0b01.
ROR	Rotate right, encoded as type = 0b11.

<Rs> The register whose bottom byte contains the amount to shift by.

cond	Mnemonic extension	Meaning (integer)	Meaning (floating-point) [a]	Condition flags
0000	EQ	Equal	Equal	Z == 1
0001	NE	Not equal	Not equal, or unordered	Z == 0
0010	CS [b]	Carry set	Greater than, equal, or unordered	C == 1
0011	CC [c]	Carry clear	Less than	C == 0
0100	MI	Minus, negative	Less than	N == 1
0101	PL	Plus, positive or zero	Greater than, equal, or unordered	N == 0
0110	VS	Overflow	Unordered	V == 1
0111	VC	No overflow	Not unordered	V == 0
1000	HI	Unsigned higher	Greater than, or unordered	C == 1 and Z == 0
1001	LS	Unsigned lower or same	Less than or equal	C == 0 or Z == 1
1010	GE	Signed greater than or equal	Greater than or equal	N == V
1011	LT	Signed less than	Less than, or unordered	N != V
1100	GT	Signed greater than	Greater than	Z == 0 and N == V
1101	LE	Signed less than or equal	Less than, equal, or unordered	Z == 1 or N != V
1110	None (AL) [d]	Always (unconditional)	Always (unconditional)	Any

2. (30 points) Complete the table by showing the contents of the registers and memory locations after each instruction that changes them (i.e., only show the contents of a register or memory location when an instruction modifies the contents), and by showing the condition codes in the non-grayed boxes. All numbers shown are in 32-bit hex values. Your entries should also be shown in 32-bit hex values.

Instruction	Register contents						Status Bits				Memory (Hex)		
	R0	R1	R2	R3	R4	R5	N	Z	C	V	00101000	00101004	00101008
Initial Conditions	000000FF	00101000	00000055	00101000	FFFFFFFF	00101008					000000FF	00000055	000000AA
mov r0, #4													
mov r4,r0													
ldr r2,[r1,r0]!													
ldr r3,[r1,r0]!													
str r4,[r5],-#4													
str r3,[r5],-#4													
ldr r2,[r3,r4, lsl #1]													
movs r0, #1													
mvns r1, r0													
adds r2,r0,r1													
adds r3,r2,r1													
subs r4,r5,r5													
subs r4,r4,#1													

3. (12 points) Each row of the table presents a number with a given magnitude. Complete the table so that each number is represented in each base.

Hex	Dec	Bin
A1		
	132	
		10101100

4. (20 points) Write ARM assembly instructions to implement the behavior documented in the C statements below. Data is stored in consecutive memory starting at 0x00100000, and the result must be stored in R0. If an overflow occurs at any time, place a 1 in R1 and terminate the loop. End in an infinite loop.

```
Y = 0;
For (I = 0; I <16; I++)
{
    if X[I] > 0
    Y = Y+X[I];
}
```

5. (20 points) Write an assembly language program to put the smallest of 20 numbers stored in consecutive memory locations starting at "dataset1" into R1. When the program is completed, enter an infinite loop.

```
.text
.global main
.equ dataset1, 0x1000

main:
```