A0

George Wang

Q1.

100	ADDI	x0,	xZR,	#12
104	ADDI	x1,	xZR,	#43
108	ADDI	x2,	xZR,	#-2
112	ADD	x4,	x0,	x1
116	ADD	x4,	x4,	x2

Q2.

36	SUB	x4, x2, x3
40	CBZ	x4, #3
44	SUB	x3, x3, x2
48	B	#2
52	ADD	x3, x3, x2
56	ADD	x4, XZR, x3

Q3.

```
40  LDUR  x4, [x4, #0]
44  STUR  x4, [x5, #64]
```

Q4. a)

40	ADD	x2, xZR, xZR	2
44	ADD	x3, xZR, xZR	2
48	ADD	x5, x4, xZR	2
52	SUBI	x6, x3, #10	2
56	CBZ	x6, #7	3
60	LDUR	x6, [x5, #0]	6
64	ADD	x2, x2, x6	2
68	ADDI	x3, x3, #1	2
72	ADDI	x5, x5, #8	2
76	SUBI	x6, x3, #10	2
80	CBNZ	x6, #-5	3

x5 is adr of a[i], x5 = x4 + i×8
x6 is tmp register to hold intermediate values.

b)

36	ADD	x2, xZR, xZR	2
40	ADDI	x3, xZR, #9	2
44	ADDI	x5, x4, #72	2
48	ADDI	x7, x4, #800	2
52	ADDI	x6, x3, #1	2
56	CBZ	x6, #8	3
60	LDUR	x6, [x5, #0]	6
64	ADD	x2, x2, x6	2
68	SUBI	x3, x3, #1	2
72	SUBI	x5, x5, #8	2
76	STUR	x2, [x4, #800]	6
80	ADDI	x6, x3, #1	2
84	CBNZ	x6, #-6	3

x3 = 9
x3 = 0
-1 = -1

x7 hold position after array a, hence x7 = x4 + 100×8
Same x5, x6 as in part a

c) part a: 3+2×4 + (6 + 4×2 + 3)×10 = 11 + 17×10 = 181 cc
part b: 3+2×5 + (6 + 2×4 + 6 + 3)×10 = 13 + 23×10 = 243 cc
part a is faster as it requires less clock cycles.

Q5.

a)

```
0   ADD    x1, xZR, xZR
4   CBZ    x2, #4
8   ADD    x1, x1, x3
12  SUBI   x2, x2, #1
16  CBNZ   x2, #-2
20  STUR   x1, [xZR, #80]
```

x_1 store sum of repeated addition.

b)

```
0   ADDI   x0, xZR, #1
4   CBZ    x15, #12
8   ADD    x0, xZR, x10
12  SUBI   x15, x15, #1
16  CBZ    x15, #9
20  SUBI   x15, x15, #1
24  ADD    x1, xZR, xZR
28  ADD    x4, xZR, x10
32  SUBI   x4, x4, #1
36  ADD    x1, x1, x0
40  CBNZ   x4, #-2
44  ADD    x0, xZR, x1
48  CBNZ   x15, #-7
52  STUR   x0, [xZR, #80]
```

x0 is storing result after each multiplication
x1 store intermediate sum result during each multiplication
x4 is the loop counter for each multiplication