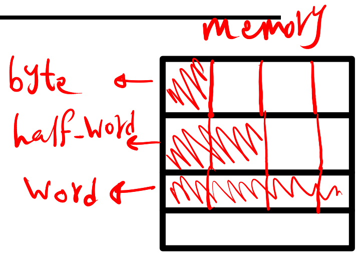
بسم الله الرحمن الرحيم

Faculty of Engineering
Computer and Systems Engineering Department

CSE 211 [Spring 2022]
Introduction to Embedded Systems

SHEET 2



Q1. What are the differences between the following three instructions?

LDRB R0, [R1] → load byte

LDRH R0, [R1] → load half-Word [2-bytes]

LDR R0, [R1] → load Word [4-bytes]

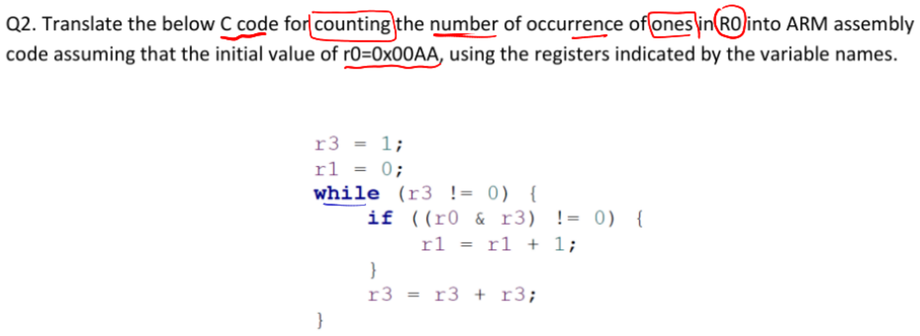
Q2. Translate the below C code for counting the number of occurrence of ones in R0 into ARM assembly code assuming that the initial value of r0=0x00AA, using the registers indicated by the variable names.

```
r3 = 1;
r1 = 0;
while (r3 != 0) {
    if ((r0 & r3) != 0) {
        r1 = r1 + 1;
    }
    r3 = r3 + r3;
}
```

Q3. Explain what an ARM processor accomplishes in terms of accessing and changing its registers when it executes a BEQ instruction.

Q4. Translate the below C code into ARM assembly code, using the registers indicated by the variable names. The C code presumes that r0 holds the address of the first entry of an array of integer values, and r1 indicates how many elements the array holds; the code removes all adjacent duplicates from the array.

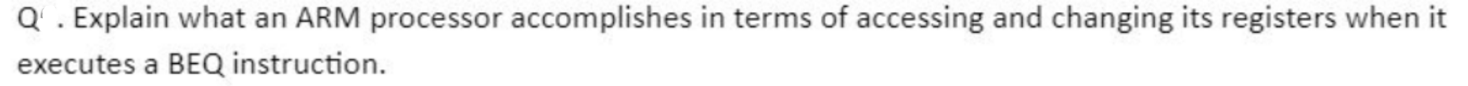
```
r3 = 1;
for (r2 = 1; r2 < r1; r2++) {
    if (r0[r2] != r0[r2 - 1]) {
        r0[r3] = r0[r2];
        r3 += 1;
    }
}
r1 = r3;
```



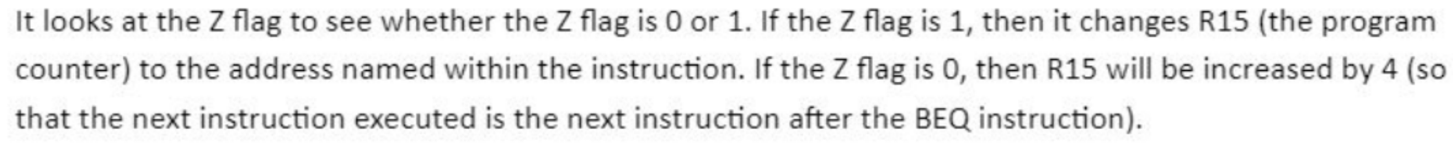
Q2. Translate the below C code for counting the number of occurrence of ones in R0 into ARM assembly code assuming that the initial value of r0=0x00AA, using the registers indicated by the variable names.

```
r3 = 1;
r1 = 0;
while (r3 != 0) {
    if ((r0 & r3) != 0) {
        r1 = r1 + 1;
    }
    r3 = r3 + r3;
}
```

```
        MOV R3,#1   ; R3 = 1
        MOV R1,#0   ;  R1 = 0
loop:   CMP R3,#0   ; is (R3 = = 0) ?
        BEQ Exit
        ANDS R0, R0, R3 ; after getting R0 Compare it with zero flag
        ; BEQ out from if
        ADDNE R1,R1,#1 ; in case condition is true execute ADDNE
; Out from if
        ADD R3,R3,R3 ; R3 = R3 + R3
        B loop
Exit:
```



Q . Explain what an ARM processor accomplishes in terms of accessing and changing its registers when it executes a BEQ instruction.



It looks at the Z flag to see whether the Z flag is 0 or 1. If the Z flag is 1, then it changes R15 (the program counter) to the address named within the instruction. If the Z flag is 0, then R15 will be increased by 4 (so that the next instruction executed is the next instruction after the BEQ instruction).

ex BEQ loop ;

Check Z-flag if

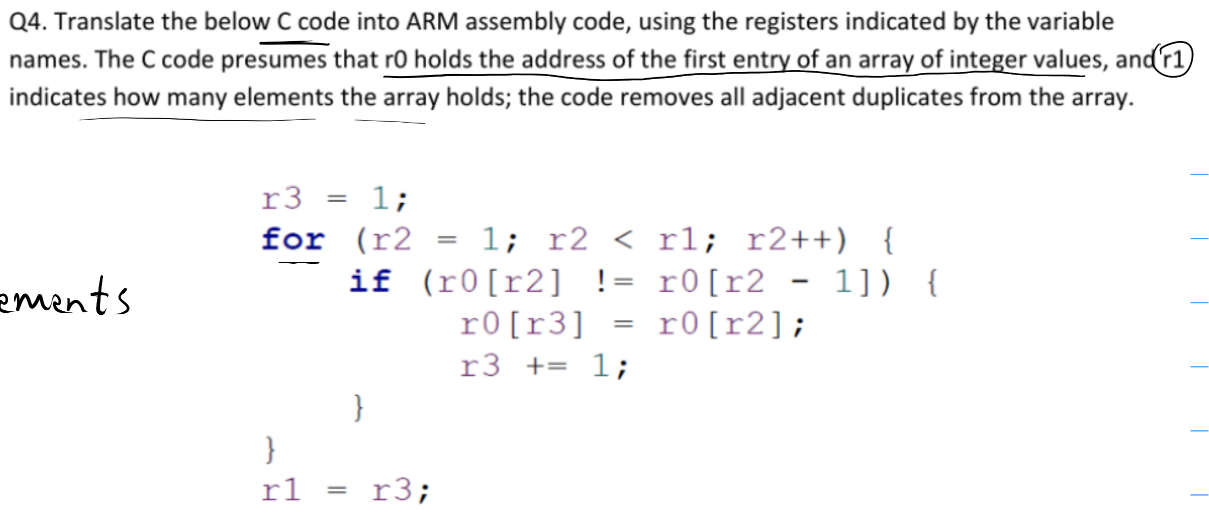
Z = 1 → means equal

∴ R15 [PC] → go to loop

if Z-flag = 0 → means not equal

∴ R15 [PC] → 'll execute next instruction PC+4

Q(4)



Q4. Translate the below C code into ARM assembly code, using the registers indicated by the variable names. The C code presumes that r0 holds the address of the first entry of an array of integer values, and r1 indicates how many elements the array holds; the code removes all adjacent duplicates from the array.

```
r3 = 1;
for (r2 = 1; r2 < r1; r2++) {
    if (r0[r2] != r0[r2 - 1]) {
        r0[r3] = r0[r2];
        r3 += 1;
    }
}
r1 = r3;
```

r0 → base register for array

r1 → no of array elements

```
            MOV  R3,#1
            Mov  R2, #1
            Mov R1,#Array_size
loop:       CMP  R2,R1
            BGE  Done    ; if R2>R1 → get out from loop
            LDR  R4, [R0,R2,LSL#2]
            ; R4→temporary ( R0→base + R2(offset) * 4
            sub  R5, R2, #1  ;R5 is an offset refer to element of array
                             ; before element pointed to by R2
            LDR  R5,[R0, R5, LSL # 2]
            → ; أنا كده جبت العنصر اللي في ال array و العنصر اللي قبله علطول علشان أقارنه بينهم

            CMP  R5,R4 ;  is (R5 == R4) ?
            BEQ  Outfromif   ;check Z-flag if "1" go to out from if
            ممكن مستعملش BEQ وأكتب العملية بعدها على طول وأستخدم Suffix
;
; في حالة عدم التساوي
            STRNE  R4,[R0,R3, LSL#2]
; تنفذ
            ADDNE  R3 ,R3 ,#1
            STR  R4,[R0 ,R3, LSL #2]
            ADD  R3,R3, #1
outfromif:  B   loop

Done:       Mov R1, R3
```

Q5. Translate the below C fragment into an equivalent ARM assembly language program, using registers corresponding to the variable names. Assume r0 and r1 hold signed values.

```
r2 = 0;
while (r1 != 0) {
     if ((r1 & 1) != 0) {
          r2 += r0;
     }
     r0 <<= 1;
     r1 >>= 1;
}
while (1); // halting loop
```

Q6. For the below ARM assembly code, trace the values that will be placed into the registers R4, R5, and R6. By tracing, you are expected to write the values of the mentioned registers after the execution of each instruction.

R4	R5	R6
7	4	4
11	7	3
18	11	2
29	18	1
47	29	0

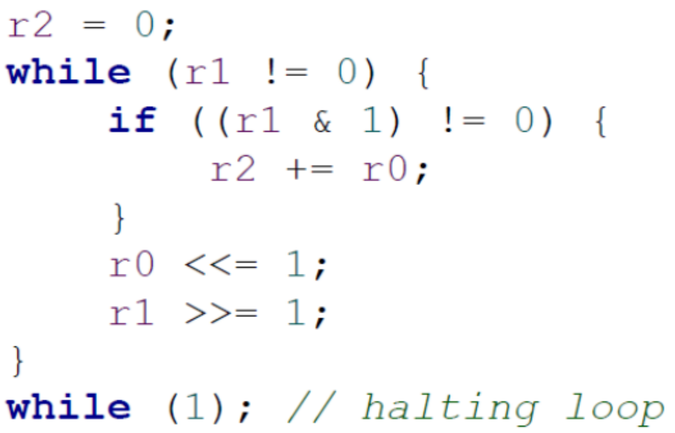
```
        MOV  R4, #7
        MOV  R5, #4
        MOV  R6, #4
again   MOV  R7, R4       R7= R4
        ADD  R4, R5, R4   R4=R4 +R5
        MOV  R5, R7       R5 = R7
        SUBS R6, R6, #1   R6= R6-1
        BNE  again
```

Q(5)

Q5. Translate the below C fragment into an equivalent ARM assembly language program, using registers corresponding to the variable names. Assume r0 and r1 hold signed values.

check lect.

annot.



```
r2 = 0;
while (r1 != 0) {
    if ((r1 & 1) != 0) {
        r2 += r0;
    }
    r0 <<= 1;
    r1 >>= 1;
}
while (1); // halting loop
```