Program Understanding 2

- Show all updates to registers as the assembly code shown below runs, assuming little-endian ordering. (Memory addresses increase from top to bottom, and from left to right in the table.) Show the NZCV flags next to the instruction if they change. Each instruction is 32 bits. Do not update PC after each instruction. R13 and R15 have initial values shown.

```
     LDR R1, =0x10000010
     LDR R2, [R1]
     LDR R3, [R1,#4]
     BL max
     SUBS R4, R2, R0  @NZCV =
     MOVW R5, #1
     LSL R6, R5, #4
     BIC R7, R2, R6
     ANDS R8, R7, R6  @NZCV =
     ROR R9, R3, #12
     REV R10, R9
     RBIT R11, R10
     ADDS R12, R3, R9 @NZCV =
     STR R12, [R1,#8]

loop B loop
      ENDP

max PROC
      CMP R2, R3       @NZCV =
      BLT second
first MOV R0, R2
       B done
second MOV R0, R3
done   BX LR
      ENDP
```

R0		R4		R8		R12	
R1		R5		R9		R13	0x100 00200
R2		R6		R10		R14	
R3		R7		R11		R15	0x000 00250

0x10000010	FF	EF	CD	AB	00	00	CD	AB	00	00	00	00

Program Understanding 2 ANS

- Step-by-step execution shown in next slide.

```
    LDR R1, =0x10000010
    LDR R2, [R1]
    LDR R3, [R1,#4]
    BL max
    SUBS R4, R2, R0  @NZCV = 0110
    MOVW R5, #1
    LSL R6, R5, #4
    BIC R7, R2, R6
    ANDS R8, R7, R6  @NZCV = 0110
    ROR R9, R3, #12
    REV R10, R9
    RBIT R11, R10
    ADDS R12, R3, R9 @NZCV = 1000
    STR R12, [R1,#8]

loop B loop
     ENDP

max PROC
     CMP R2, R3      @NZCV = 0010
     BLT second
first MOV R0, R2
```

R0	0xABCDEFFF	R4	0x00000000	R8	0x00000000	R12	0xABD7BCD0
R1	0x10000010	R5	0x00000001	R9	0x000ABCD0	R13	0x10000200
R2	0xABCDEFFF	R6	0x00000010	R10	0xD0BC0A00	R14	0x00000260
R3	0xABCD0000	R7	0xABCDEFEF	R11	0x00503D0B	R15	omitted

0x10000010	FF	EF	CD	AB	00	00	CD	AB	D0	BC	D7	AB

- LDR R1, =0x10000010 → R1 = 0x10000010 (literal load of the base address).
- LDR R2, [R1] → reads bytes FF EF CD AB at 0x10000010..13 and forms R2 = 0xABCDEFFF (little-endian).
- LDR R3, [R1,#4] → reads bytes 00 00 CD AB at 0x10000014..17 and forms R3 = 0xABCD0000 (little-endian).
- BL max → in max: CMP R2, R3 sets NZCV = 0010 since R2 > R3 (N=0, Z=0, C=1,V=0), path takes first: MOV R0, R2, then BX LR returns with R0 = 0xABCDEFFF.
- SUBS R4, R2, R0 → R4 = R2 − R0 = 0, NZCV = 0110 (N=0, Z=1, C=1,V=0) because subtracting equal values yields zero with carry set.
- MOVW R5, #1 → R5 = 0x00000001 (loads 16-bit immediate into low halfword).
- LSL R6, R5, #4 → R6 = 0x00000010 (1 shifted left by 4).
- BIC R7, R2, R6 → R7 = R2 & ~R6 = 0xABCDEFFF & 0xFFFFFFEF = 0xABCDEFEF (clears bit 4).
- ANDS R8, R7, R6 → R8 = 0xABCDEFEF & 0x10 = 0x00000000 with NZCV = 0110 (zero result with carry preserved from previous SUBS instruction).
- ROR R9, R3, #12 → rotates 0xABCD0000 right by 12 bits to R9 = 0x000ABCD0.
- REV R10, R9 → byte-reverse 00 0A BC D0 into D0 BC 0A 00, giving R10 = 0xD0BC0A00.
- RBIT R11, R10 → bit-reverse all 32 bits of 0xD0BC0A00, yielding R11 = 0x00503D0B (per architectural bit-reverse).
- ADDS R12, R3, R9 → R12 = 0xABCD0000 + 0x000ABCD0 = 0xABD7BCD0 with NZCV = 1000 (negative due to high bit, non-zero, no carry, no overflow).
- STR R12, [R1,#8] → stores 0xABD7BCD0 at 0x10000018 as bytes D0 BC D7 AB in little-endian.

R0	0xABCDEFFF	R4	0x00000000	R8	0x00000000	R12	0xABD7BCD0
R1	0x10000010	R5	0x00000001	R9	0x000ABCD0	R13	0x10000200
R2	0xABCDEFFF	R6	0x00000010	R10	0xD0BC0A00	R14	0x00000260
R3	0xABCD0000	R7	0xABCDEFEF	R11	0x00503D0B	R15	omitted

0x10000010	FF	EF	CD	AB	00	00	CD	AB	D0	BC	D7	AB

Notes on Flags

- CMP R2,R3 → NZCV = 0010 because R2 > R3 in unsigned comparison semantics used by CMP on 32-bit registers without carry-in.
- SUBS R4,R2,R0 → NZCV = 0110 because the result is zero and subtraction of equal values sets carry and clears negative and overflow.
- ANDS R8,R7,R6 → NZCV = 0110 because the logical AND produced zero, and the carry is preserved from SUBS.
- ADDS R12,R3,R9 → NZCV = 1000 because the sum has the top bit set (negative in signed sense), is non-zero, and does not generate carry or overflow for these operands.

R0	0xABCDEFFF	R4	0x00000000	R8	0x00000000	R12	0xABD7BCD0
R1	0x10000010	R5	0x00000001	R9	0x000ABCD0	R13	0x10000200
R2	0xABCDEFFF	R6	0x00000010	R10	0xD0BC0A00	R14	0x00000260
R3	0xABCD0000	R7	0xABCDEFEF	R11	0x00503D0B	R15	omitted

0x10000010	FF	EF	CD	AB	00	00	CD	AB	D0	BC	D7	AB

R13, R14, R15

- R13 is the stack pointer (SP), R14 is the link register (LR), and R15 is the program counter (PC). Table shows
- R13 (SP): points to the current top of the stack used for pushes, pops, and call frames; here it's initialized to 0x10000200 and remains the same since there is no stack operation (PUSH, POP).
- R15 (PC): Initially, PC = 0x00000250 at the first instruction "LDR R1, =0x10000010". It gets incremented by 4 after execution of each instruction without branch, hence PC = 0x0000025C at instruction "BL max". When the program finishes, PC is at the last instruction "loop B loop", and shown as "omitted" in the table.
- R14 (LR): holds the return address set by BL (address of the instruction after the branch), which is 0x0000025C + 4 = 0x00000260.

0x00000250
0x0000025C
0x00000260

```
     LDR R1, =0x10000010
     LDR R2, [R1]
     LDR R3, [R1,#4]
     BL max
     SUBS R4, R2, R0  @NZCV =
     MOVW R5, #1
     LSL R6, R5, #4
     BIC R7, R2, R6
     ANDS R8, R7, R6  @NZCV =
     ROR R9, R3, #12
     REV R10, R9
     RBIT R11, R10
     ADDS R12, R3, R9 @NZCV =
     STR R12, [R1,#8]

loop B loop
     ENDP

max PROC
     CMP R2, R3       @NZCV =
     BLT second
first MOV R0, R2
```

R0	0xABCDEFFF	R4	0x00000000	R8	0x00000000	R12	0xABD7BCD0
R1	0x10000010	R5	0x00000001	R9	0x000ABCD0	R13	0x10000200
R2	0xABCDEFFF	R6	0x00000010	R10	0xD0BC0A00	R14	0x00000260
R3	0xABCD0000	R7	0xABCDEFEF	R11	0x00503D0B	R15	omitted

0x10000010	FF	EF	CD	AB	00	00	CD	AB	D0	BC	D7	AB

6. *Assembly*

(a) Show all updates to registers as the assembly code shown below runs. Show the NZCV flags next to the instruction if they change. Do not update PC each time. You can assume that each instruction is 32 bits.

```
0x00000250      LDR R1, =0x10000010
0x00000254      LDR R2, [R1]
0x00000258      LDR R3, [R1,#4]
0x0000025C      BL max
0x00000260      SUBS R4, R2, R0     NZCV = 0110
                MOVW R5, #1
                LSL R6, R5, #4
                BIC R7, R2, R6
                ANDS R8, R7, R6     NZCV = 0110
                ROR R9, R3, #12
                REV R10, R9
                RBIT R11, R10
                ADDS R12, R3, R9    NZCV = 1000
                STR R12, [R1,#8]
loop B loop
                ENDP

max   PROC
                CMP R2, R3          NZCV = 0010
                BLT second
first           MOV R0, R2
                B done
second          MOV R0, R3
done BX LR
                ENDP
```

EC361 2020 Exam 2 Q1
https://www.youtube.com/watch?v=XkjC1avu6Ow

R0	0xABCDEFFF	R4	0x00000000	R8	0x00000000	R12	0xABD7BCD0
R1	0x10000010	R5	0x00000001	R9	0xD0ABCD0	R13	0x10000200
R2	0xABCDEFFF	R6	0x00000010	R10	0xD0BCDA00	R14	0x00000260
R3	0xABCD0000	R7	0xABCDEFEF	R11	0x0050300B	R15	0x00000250

0x10000010	FF	EF	CD	AB	00	00	CD	AB	D0	BC	D7	A8