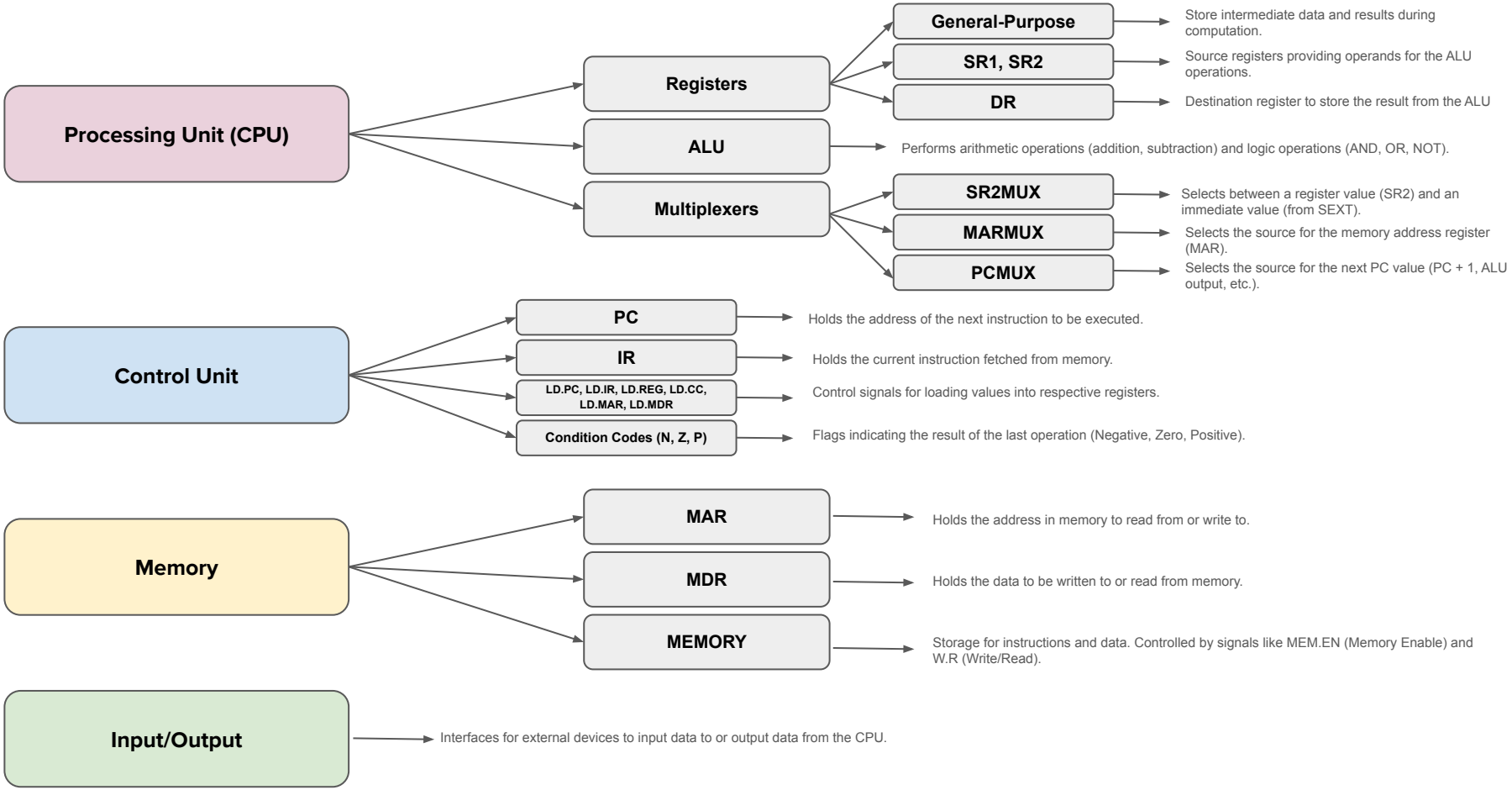
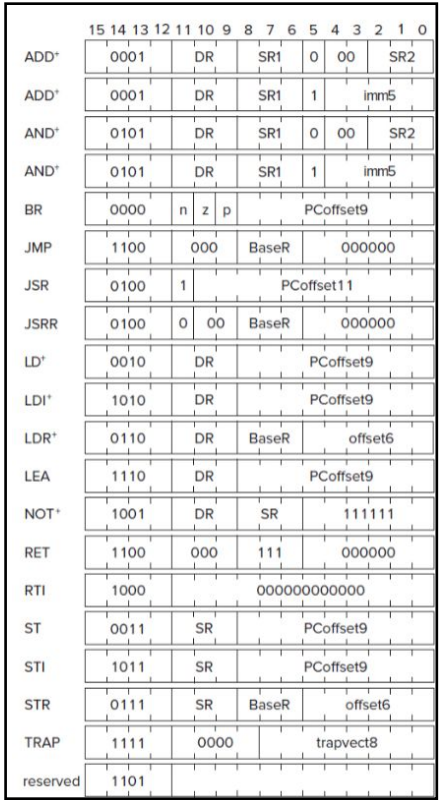

Processing Unit (CPU)
Registers
General-Purpose
Store intermediate data and results during computation.
SR1, SR2
Source registers providing operands for the ALU operations.
DR
Destination register to store the result from the ALU
ALU
Performs arithmetic operations (addition, subtraction) and logic operations (AND, OR, NOT).
Multiplexers
SR2MUX
Selects between a register value (SR2) and an immediate value (from SEXT).
MARMUX
Selects the source for the memory address register (MAR).
PCMUX
Selects the source for the next PC value (PC + 1, ALU output, etc.).
Control Unit
PC
Holds the address of the next instruction to be executed.
IR
Holds the current instruction fetched from memory.
LD.PC, LD.IR, LD.REG, LD.CC, LD.MAR, LD.MDR
Control signals for loading values into respective registers.
Condition Codes (N, Z, P)
Flags indicating the result of the last operation (Negative, Zero, Positive).
Memory
MAR
Holds the address in memory to read from or write to.
MDR
Holds the data to be written to or read from memory.
MEMORY
Storage for instructions and data. Controlled by signals like MEM.EN (Memory Enable) and W.R (Write/Read).
Input/Output
Interfaces for external devices to input data to or output data from the CPU.

ADD R6, R3, R7	ADD (+) whatever is in R3 with whatever is in R7 and store this in R6
AND R4, R0, R5	BITWISE AND () whatever is in R0 and R5, store the result in R4
ADD R6, R3, #-1	ADD (+) whatever is in R3 with #number, store the result in R6
AND R4, R0, #3	BITWISE AND () whatever is in R0 with #number, store the result in R4
NOT R0, R2	BITWISE NOT whatever is in R2, store the result in R0
LD R1, #2	PC + #number = address, load content of address in R1
ST R1, #15	Content of address in R1 is stored in address PC + #number
LDI R1, #13	PC + #number = address, read whatever is stored in this address, and use this as the address to load the data in R1.
STI R2, #13	Whatever is stored in R2, use this address to access the next address. Store this in PC + #number
LDR R4, R1, #3	R1 + #3 to get an address. Content of this address will be loaded into R4.
STR R2, R1, #3	Get whatever is in R2, and store this in R1 + #3
LEA R1, #9	Get PC + #9, store in R1
BRz x0D9	Get PC + x0D9 = x4104. If Z set to 1, then PC=x4104, else, PC=PC



MCQ 2

What should be the value of x3012 be if the following program is executed

Program:

.ORIG	x3000 ; The first instruction is at x3000		
LD	R1, #2	Address = 3001+2 = x3003:x340F	R1=x340F
LD	R2, #0	Address = 3002+0 = x3002:x320F	R2=x320F
ST	R1, #15	Address = 3003+F = x3012	x3012: x340F
ST	R2, #15	Address = 3004+F = x3013	x3013: x320F

Initial Memory Status:

Address: Content

x3000: x2202

x3001: x2400

x3002: x320F

x3003: x340F

Starting from x3004, all of them are x0000

What is the machine code of the instruction "LD R1, 9"?

0010 0010 0000 1001 = x2209

What is the machine code of the instruction "ADD R3, R3, R2" in hexadecimal?

0001 0110 1100 0010 = x16C2

What is the effect of executing instruction 0101100101101111? It should be assumed that the instruction is at address x300c.

0101 100 101 1 01111 = AND R4 R5 #15

R5 = x5678 AND x000F = x0008. The value in R4 is set to x0008

What is the operation of the LD instruction?

DR = M[PC + SEXT(PCoffset9)]

What should be the value of R1 if the following program is executed (based on the initial register status)?

Program:

AND R3, R3, #0	R3 = x0012 AND x0000	R3 = x0000
NOT R0, R2	R2 = x0003 NOT	R0 = xFFFC
ADD R1, R0, R0	R0 = xFFF8 ADD xFFF8	R1 = xFFF8
AND R3, R3, R2	R2 = x0003 AND x0000	R3 = x0000
ADD R1, R1, #-1	R1 = xFFF8 - x0001	R1 = FFF7

Initial Registers Status:

R0: x0028, xFFFC

R1: x0006, FFF8, FFF7

R2: x0003

R3: x0012, x0000, x0000

What should be the value at register R2 if the following program is executed (based on the initial memory status)?

Program:

.ORIG	x3000 ; The first instruction is at x3000		
LD	R1, #2	Address = 3001+2 = x3003:x340F	R1 = x340F
LD	R2, #0	Address = 3002+0 = x3002:x320F	R2 = x320F
ST	R1, #15	Address = 3003+15 = x3012	x3012: x340F
ST	R2, #15	Address = 3004+15 = x3013	x3013: x320F
LDI	R1, #13	Address = 3005+13 = x3012:x340F	R1 = x0000
LDI	R2, #13	Address = 3006+13 = x3013:x320F	R2 = x0000

Initial Memory Status:

Address: Content

x3000: x2202

x3001: x2400

x3002: x320F

x3003: x340F

x3004: xA20D

x3005: xA40D

Starting from x3006, all of them are x0000

LD: PC + #number = address, load content of address in R1

ST: Content of address in R1 is stored in address PC + #number

LDI: PC + #number = address, read whatever is stored in this address, and use this as the address to load the data in R1.

STI: Whatever is stored in R2, use this address to access the next address. Store this in PC + #number

x3000
x3001
x3002
x3003
x3004
x3005
x3006
x3007
x3008
x3009
x300A
x300B
x300C
x300D
x300E
x300F
x3011
x3012
x3013
x3014
x3015
x3016
x3017
x3018
x3019
x301A
x301B
...
x3020
x3021
x3022
...

MCQ 3

What should be the value of R3 if the following program is executed (based on the initial register status)?

Program:
.ORIG x3000 ; The first instruction is at x3000
AND R1, R1, #0 R1 = x0000
AND R0, R0, #0 R0 = x0000
AND R2, R2, #0 R2 = x0000
LEA R4, #5 Address = 3004 + 5 = x3009 R1 = x3009
LEA R0, #-4 Address = 3005 - 4 = x3001 R0 = x3001
LEA R2, #8 Address = 3006 + 8 = x300E R2 = x300E
LEA R3, #-9 Address = 3007 - 9 = x2FFE R3 = x2FFE
LEA R1, #9 Address = 3008 + 9 = x3011 R1 = x3011
Initial Registers Status:
R0: x0000, x0000, x3001
R1: x0000, x0000, x3009, x3011
R2: x0000, x0000, x300E
R3: x0000, x30F8
R4: x0000

Which one of the following statements is correct regarding the execution of instruction 1100000101000000? It should be assumed that the instruction is at address x3008.
1100 000 101 000000 = JMP R5 (R5 is at x3001), The address of the instruction that will be executed after this instruction is x3001.

What is the machine code of the instruction "BRnzp LOOP" according to the following program?

Program:

```
        .ORIG   x3000
        LEA     R1, INTE    ; R1 is the location of the integer
        AND     R3, R3, 0   ; R3 is the sum
        AND     R2, R2, 0   ; R2 is the count of the integers
        ADD     R2, R2, 12
LOOP    BRz EXIT
        LDR     R4, R1, 0
        ADD     R3, R3, R4
        ADD     R1, R1, 1
        ADD     R2, R2, -1
        BRnzp   LOOP
EXIT    TRAP    x25         ; halt
INTE    .BLKW   12
        .END
```

BRnzp is at x3009, So PC = x300A
LOOP is at x3004
So offset value = x300A - x3004 = #-6 (b1010)
0000 1111 1111 1010 = x0FFA

LEA: Get PC + #9, store in R1
LDI: PC + #number = address, read whatever is stored in this address, and use this as the address to load the data in R1.

What should be the value of R3 if the following program is executed?

Program:

```
        .ORIG x3000
x3000   LEA R1, INTE  ; R1 is the location of the integer
x3001   AND R2, R2, 0 ; R2 is the count of the integers
x3002   ADD R2, R2, 3
x3003LOOP ADD R4, R2, 0
x3004   STR R4, R1, 0
x3005   ADD R1, R1, 1
x3006   ADD R2, R2, -1
x3007   BRp LOOP
x3008   LEA R1, INTE  ; R1 is the location of the integer
x3009   AND R3, R3, 0 ; R3 is the sum
x300A   AND R2, R2, 0 ; R2 is the count of the integers
x300B   ADD R2, R2, 3
x300CLOOP1 LDR R4, R1, 0
x300D   ADD R3, R3, R4
x300E   ADD R1, R1, 1
x300F   ADD R2, R2, -1
x30AA   BRp LOOP1
x3012EXIT TRAP 37     ; halt
x3013INTE .BLKW 12
        .END
```

R1: x3016
R2: x0000
R3: x0006
R4: x0001

x3013: x0003
x3014: x0002
x3015: x0001

LDR R4, R1, #3	R1 + #3 to get an address. Content of this address will be loaded into R4.
STR R2, R1, #3	Get whatever is in R2, and store this in R1 + #3

```
        .ORIG X3000
        LD    R0, A
        LEA   R1, B
        LDR   R2, R1, #-1
        LDI   R3, C
        AND   R4, R3, #7
        STR   R4, R1, #0
        STI   R4, D
        HALT
A       .FILL X1234
        .FILL X2345
B       .BLKW 2
C       .FILL X3008
D       .FILL X300B
        .END
```

For the program above, which one of the following statements is correct?
"LDI R3, C"

x3008: x1234
x3009: x2345
x300A:
x300B
x300C: x3008
x300D: x300B

Address of 'C' = x300C, So x300C stores x3008, which stores x1234. Hence x1234 is stored into R3.

Instruction "LDI R3, C" sets the value of R3 to x1234.

What is the condition code after the execution of "NOT R4, R2" based on the following program?

Program:

```
        .ORIG x3000
        LD R3, V3       ; Load V3 to R3
        LD R2, V2       ; Load V2 to R2
        LD R1, V1       ; Load V1 to R1
        LD R0, V0       ; Load V0 to R0
        ADD R4, R1, R3
        NOT R4, R2                  1111 1111 1111 1110 → 0000 0000 0000 0001 → x0001 (Positive)
        AND R4, R2, R1
        LDR R4, R0, 5
EXIT    TRAP x25        ; halt
V0      .FILL x3000     ; M[V0] = x3000
V1      .FILL x0001     ; M[V1] = 1
V2      .FILL xFFFE     ; M[V2] = -2
V3      .FILL xFFFF     ; M[V3] = -1
        .END
```

R0: x3000
R1: x0001
R2: xFFFE
R3: xFFFF
R4: x0001

What is the machine code of the instruction "LEA R0, -8"?
1110 0001 1111 1000 = xE1F8

What should be the value of R1 if the following program is executed (based on the initial register status)?

Program:

```
.ORIG       x3000 ; The first instruction is at x3000
AND R0, R0, #0
ADD R2, R0, #2
ADD R3, R0, #3
LEA R1, #9              PC = x3004 + #9 = x300D
STR R3, R1, #3          x300D + 3 = x3010
STR R2, R1, #4          x300D + 4 = x3011
LDR R2, R1, #3          x3010: x0003 into R2
LDR R3, R1, #4          x3011: x0002 into R3
```

Initial Registers Status:
R0: x0000
R1: x0000
R2: x0000
R3: x0000
R4: x0000

R0: x0000
R1: x300D
R2: x0003
R3: x0002

x3010: x0003
x3011: x0002

MCQ 4

What will be printed out after the execution of the following program if the values of "VALA" (mem[VALA]) and "VALB" (mem[VALB]) are set to x0004 and 0x0005 respectively at the beginning?

Program:

```
        .ORIG x3000
        AND R0, R0, #0
        LD R1, VALA
        LD R2, VALB
        NOT R2, R2              1111 1111 1111 1010 = xFFFA
        ADD R2, R2, #1
        ADD R3, R1, R2          x0004 + FFFB = xFFFF
        BRn BIGB ; if VALB > VALA
        LD R0, VALB
        BRnzp DISP
BIGB    LD R0, VALA
DISP    LD R1, OUT0
        ADD R0, R0, R1
        OUT
EXIT    HALT
VALA    .BLKW1      x0004
VALB    .BLKW1      x0005
OUT0    .FILLx30 ;ASCII of '0'
        .END
```

R0: '4'
R1: '0'
R2: x0005, xFFFA, xFFFB
R3: xFFFF
Output: 4

What is the output of the program if '1' is inputted? Given that the ASCII code of 'a' is x61 and '0' is x30.

Program:

```
        .ORIG x3000
        LD R2, TERM ;
        LD R3, ASCII ; Load ASCII difference
AGAIN   TRAP x23 ; input character
        ADD R1, R2, R0 ; Test for terminate
        BRz EXIT ; Exit if done
        ADD R0, R0, R3
        TRAP x21 ; Output to monitor...
        BRnzp AGAIN
TERM    .FILL xFFc9
ASCII   .FILL x0031
EXIT    TRAP x25 ; halt
        .END
```

R0: x0061
R1: xFFFA
R2: xFFC9
R3: x0031
Output: b

```
        .ORIG X3100
        LD    R0, A
D       LEA   R1, B
        ST    R0, C
        HALT
A       .STRINGZ "XYZ"
B       .BLKW 5
C       .FILL 0
        .END
```

For the program above, which one of the statements below is correct?

a. The value stored at memory location with label A is the ASCII code of the string "XYZ".

The memory location A points to the first letter in the string 'X' represented by its ASCII value: x0058.

b. The value stored at memory location with label B must be 0.

B reserves five blank memory locations all set to x0000

c. The value stored at memory location with label B is 5.

B reserves five blank memory locations all set to x0000

d. The value stored at memory location with label A is the ASCII code of character "X".

The memory location A points to the first letter in the string 'X' represented by its ASCII value: x0058.

What is the output of the program if 'Z' is inputted? Given that the ASCII code of 'a' is x61 and 'A' is x41.

Program:

```
        .ORIG x3000
        LD R2, TERM
        LD R3, ASCII ; Load ASCII difference
AGAIN   TRAP x23 ; input character
        ADD R1, R2, R0 ; Test for terminate
        BRz EXIT ; Exit if done
        ADD R0, R0, R3
        TRAP x21 ; Output to monitor...
        BRnzp AGAIN
TERM    .FILL xFFc9
ASCII   .FILL x0020
EXIT    TRAP x25 ; halt
        .END
```

R0: Z (x005A), x007A
R1: x0023
R2: xFFC9
R3: x0020
Output = x007A (z)

What will be printed out after the execution of the following program?

Program:

```
        .ORIG x3000
        LEA R1, hello  ; R1 points to the character
        LD R0, VAL
        ADD R0, R1, R0          ;
        LDR R0, R0, #0          ; R0 holds the character
        TRAP x21                ; or just OUT prints R0[7:0]
        TRAP x25                ; or HALT
hello   .STRINGZ "Hello world"
VAL     .FILL #7
        .END
```

R1: 'o'
R0: x0007

What is the value of R3 after the execution of the following program if the value of "NUM" (mem[NUM]) is set to x0002 at the beginning?

Program:

```
        .ORIGx3000
        LD R1, SIX
        LD R2, NUM
        AND R3, R3, #0
; The inner loop
;
AGAIN   ADD R3, R3, R2
        ADD R1, R1, #-1
        BRzp AGAIN
;
        HALT
;
NUM     .BLKW #1 x0002
SIX     .FILL x0006
        .END
```

R1: xFFFF
R2: x0002
R3: x000E
R3 holds x000E.

MCQ 5

What is the value of the register R3 after the execution of the following program when "VALN" is 5?

Program:

```
            .ORIG x3000
            LD R2, VALN
            ADD  R4, R2, #0
            ADD  R1, R2, #-1
; Outer Loop
OLoop       AND R3, R3, #0
; Inner Loop
ILoop       ADD R3, R3, R4
            ADD R1, R1, #-1
            BRp ILoop
; End of inner Loop
            ADD R4, R3, #0
            ADD R2, R2, #-1
            ADD R1, R2, #-1
            BRp OLoop
; End of outer Loop
            HALT
VALN        .BLKW 1 #5
            .END
```

R1: x0000
R2: x0001
R3: x0078
R4: x0078
At the end of the program, R3 = x0078

For the program above, what is the machine code of instruction "LDR R0, R2, #2"?

```
        .ORIG X3000
        ADD   R7, R6, #-5
        LEA   R6, Y
        BRz   B
A       LDR   R0, R2, #2
B       HALT
C       .BLKW       5
X       .FILL  2
Y       .FILL  1
Z       .FILL  2
        .END
```

0110 0000 1000 0010 = x6082

The hexadecimal representation of a machine instruction is x5543. The address of the instruction is x4000. Which one of the assembly instructions below corresponds to the machine instruction?

a. AND R2, R5, R3
b. AND R2, R5, #3
c. ADD R2, R5, #3
d. ADD R2, R5, R3
e. AND R2, R5, R4

x5543 = 0101 010 101 000 011 = AND R2, R5, R3

What is the machine code of the instruction "BRz finish"?

Program:

```
            .ORIG x3000
            LEA R1, hello
loop        LDR R0, R1, #0
            BRz finish
            TRAP x21
            ADD R1, R1, #1
            BRnzp loop
finish      TRAP x25
hello       .STRINGZ "Hello world"
            .END
```

BRz = 0000 010
PC = x3003, finish = x3006, Offset = 3
0000 0100 0000 0011 = x0403

What is the value of R3 after the execution of the following program if the values of "VALA" (mem[VALA]) and "VALB" (mem[VALB]) are set to x0000 and 0x0005 respectively at the beginning?

Program:

```
            .ORIG x3000
            AND R3, R3, #0
            LD R2, VALB
            LD R1, VALA
; Loop
AGAIN       BRnz EXIT
            ADD R3, R3, R2
            ADD R1, R1, #-1
            BRnzp AGAIN
;
EXIT        HALT
VALA        .BLKW 1 x0000
VALB        .BLKW 1; x0005
            .END
```

R3: x0000
R2: x0005
R1: x0000

The value stored in R3 will be x000 after the execution of this program.

What is the value passed to the OS stack during the execution of the instruction "OUT" based on the following program?

Program:

```
            .ORIG x3000
            LD R2, TERM
            LD R3, ASCII
AGAIN       TRAP x20
            ADD R1, R2, R0
            BRz EXIT
            OUT
            ADD R0, R0, R3
            TRAP x21
            BRnzp AGAIN
TERM        .FILL xFFC9
ASCII       .FILL x0020
EXIT        TRAP x25      ; halt
            .END
```

R0: x3003
R2: xFFC9
R3: x0020

OS STACK = [x3003]

When a trap is called, the incremented PC is put onto the OS stack. Then the PC will perform the TRAP. When it is done, it pops off the value on the OS stack and puts that into the PC to get back to the original program. This is stored in R0.

TRAP and OUT both use the OS Stack.

MCQ 7

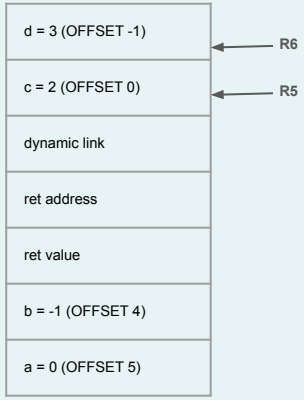
Read the program below before answering the questions 2 to 9.

```
int a=0;
int b=-1;

int main() {
  int c=2;
  int d=3;
  if (d < c)
    a = c++ ;
  else
    d = d - b;
}
```

Assume that (i) rG represents the register pointing to the beginning of the locations where the global variables are stored, (ii) rL denotes the register that records the start of the locations where the local variables are stored, and (iii) the offsets of the variables in the program above are given in the table below. In the table, each capital letter represents an integer.

variable	offset
a	A
b	B
c	C
d	D



Which one of the following LC-3 instruction sequences will be generated when converting statement "int a=0;" to LC-3 assembly language instructions?

and rG, rG, 0
str rG, rG, A

and rL, r0, 0
str rL, rL, A

and r0, r0, 0
str r0, rL, A

and r0, r0, 0
str r0, rG, A

Which one of the following LC-3 instruction sequences will be generated when converting statement "int c=2;" to LC-3 assembly language instructions?

and r0, r0, 0
add r0, r1, 2
str r0, rL, C

and r0, r0, 0
add r0, r0, 2
str r0, rG, C

and r0, r1, 0
add r0, r0, 2
str r0, rL, C

str #2, rL, C

Which one of the following LC-3 instruction sequences will be generated when converting statement "int d=3;" to LC-3 assembly language instructions?

and r0, r0, 0
add r0, r0, 3
str r0, rG, D

add r0, r0, 3
str r0, rL, D

str #3, rL, D

and r0, r0, 0
add r0, r0, 3
str r0, rL, D

Which one of the following LC-3 instruction sequences will be generated when converting statement "a = c++;" (i.e., the if branch of the "if … else …" statement) to LC-3 assembly language instructions? Label "next" should be regarded as the label of the statement that follows the "if … else …" statement.

ldr r0, rL, C
add r0, r0, 1
str r0, rL, C
str r0, rG, A
brnzp next

dr r0, rG, C
str r0, rK, A
add r0, r0, 1
str r0, rG, C

ldr r0, rL, C
str r0, rG, A
add r0, r0, 1
str r0, rL, C
brnzp next

ldr r0, rG, C
str r0, rK, A
add r0, r0, 1
str r0, rG, C
brnzp next

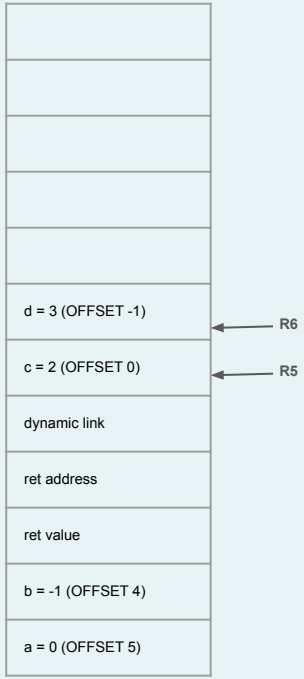
Read the program below before answering the questions 2 to 9.

```
int a=0;
int b=-1;

int main() {
  int c=2;
  int d=3;
  if (d < c)
    a = c++ ;
  else
    d = d - b;
}
```

Assume that (i) rG represents the register pointing to the beginning of the locations where the global variables are stored, (ii) rL denotes the register that records the start of the locations where the local variables are stored, and (iii) the offsets of the variables in the program above are given in the table below. In the table, each capital letter represents an integer.

variable	offset
a	A
b	B
c	C
d	D



Which one of the following LC-3 instruction sequences will be generated when converting statement "d = d - b;" (i.e. the else branch of the "if … else …" statement) to LC-3 assembly language instructions?

ldr r0, rL, D
ldr r1, rG, B
not r1, r1
add r1, r1, -1
add r0, r0, r1
str r0, rL, D

ldr r0, rL, D
ldr r1, rG, B
add r0, r0, -r1
str r0, rL, D

ldr r0, rL, D
ldr r1, rL, B
not r1, r1
add r1, r1, -1
add r0, r0, r1
str r0, rL, D

ldr r0, rL, D
ldr r1, rG, B
not r1, r1
add r1, r1, 1
add r0, r0, r1
str r0, rL, D

Which one of the following LC-3 instruction sequences will be generated when converting condition "d < c" of the "if … else …" statement to LC-3 assembly language instructions? Label "else" should be regarded as the label of the statement in the "else" branch.

ldr r0, rL, D
ldr r1, rL, C
not r1, r1
add r1, r1, 1
add r0, r0, r1
brnp else

ldr r0, rL, D
ldr r1, rL, C
not r1, r1
add r1, r1, 1
add r0, r0, r1
brn else

ldr r0, rL, D
ldr r1, rL, C
not r1, r1
add r1, r1, 1
add r0, r0, r1
brzp else

ldr r0, rG, D
ldr r1, rG, C
not r1, r1
add r1, r1, 1
add r0, r0, r1
brn else

If d < c, after making c negative, when adding d and c, the result should be negative. The if statement runs when N is set to 1, P and Z are set to 0 (ELSE).

2023 S2 MIDTERM

Convert the given decimal numbers into 9-bit unsigned or signed binary integers. If the binary integers are signed, provide your answer using the specified representation: sign magnitude, 1's complement or 2's complement signed integers.

Represent 147 as a 9-bit unsigned binary integer 010010011
Represent 106 as a 9-bit sign magnitude binary integer 001101010
Represent -53 as a 9-bit 1's complement binary integer 111000010
Represent -157 as a 9-bit 2's complement binary integer 101100001

Which of the following additions result in overflow? You can assume that both operands and the result are 8-bit 2's complement signed binary integers.

Select one or more of the following options.

00100101 + 01001111 = 37 + 79 = 116
10111111 + 11000000 = -65 + -64 = -129 (overflow)
11100111 + 11110101= -25 - 11 = -36
01000011 + 00111111 = 67 + 63 = 130 (overflow)

For an 8-bit 2's Complement, -2^{n-1} to $2^{n-1} - 1$ values are represented (-2^8 to $2^8 - 1$)
Range = -128 to 127

The LC-3 architecture uses 16-bit addresses and has an addressability of 16 bits. Given this information, answer the following questions:

How much memory, in bytes, does the LC-3 architecture have available?

If the memory is arranged as a one dimensional array of memory locations, how many outputs would the memory address decoder need?

If the memory is arranged as a two dimensional grid of memory locations, requiring a row and column address decoder, what would the total number of outputs be (row decoder outputs + column decoder outputs)?

What is the assembly language code for the instruction represented by x56BA?
0101 011 010 1 11010 = AND R3 R2 #-26

The initial value of the registers are:
R0: x0001; R1: xAAAA; R2: x5555; R3: x00F7;
R4: x0000; R5: xFFC1; R6: x003F; R7: x0000;

(a) What is the value of R0 after AND R0, R1, R2 is executed?
xAAAA AND x5555
0101 0101 0101 0101 AND 1010 1010 1010 1010 = x0000

(b) What is the value of R3 after NOT R3, R3 is executed?
NOT x00F7
0000 0000 1111 0111
1111 1111 0000 1000 = xFF08

(c) What is the value of R4 after ADD R4, R5, R6 is executed?
FFC1 ADD 003F = x0000

```
        .ORIG x3000
        LD R0, STOP
        NOT R0, R0
        ADD R0, R0, #1
        AND R3, R3, #0
        AND R4, R4, #0
        ADD R4, R4, #1
        AND R5, R5, #0
        LD  R1, COUNT
LOOP    ADD R2, R0, R1
        BRz EXIT
        ADD R5, R4, R4
        ADD R5, R5, R3
        ADD R3, R4, #0
        ADD R4, R5, #0
        ADD R1, R1, #1
        BRnzp   LOOP
EXIT    HALT
COUNT   .FILL   x0002
STOP    .FILL   x0005
        .END
```

Answer the following questions. All of your answers must be in hexadecimal, in the format "x" followed by 4 hexadecimal digits. For example, x1234 or xFFFF, are in the required format. Answers are not case sensitive.

What will be the value in register R5 when this program is executed?

R0	R1	R2	R3	R4	R5
xFFFB	0005	0000	0005	000C	000C

After the execution, R5 = x000C

What is the machine code for the instruction LD R0, STOP?
0010 0000 0000 0101 = x2005
What is the machine code for the instruction BRnzp LOOP?
Offset = #-6
0000 1111 1111 1010 = x0FFA

```
        .ORIG x3000
        LEA R1, NEW_TEXT
        LEA R2, TEXT
LOOP    LDR R3, R2, #0
        BRz EXIT
        ADD R3, R3, #6
        STR R3, R1, #0
        ADD R1, R1, #1
        ADD R2, R2, #1
        BRnzp   LOOP
EXIT    LEA R0, NEW_TEXT
        PUTS
        HALT
TEXT    .STRINGZ "TEST"
NEW_TEXT    .BLKW   #4
END     .FILL   x0000
        .END
```

Complete the symbol table for this program

LOOP	x3003
EXIT	x300A
TEXT	x300D
NEW_ TEXT	x3012
END	x3016

What is the output of this program?
R1: x0004
R2: NULL
R3: Z

x300D: T
x300E: E
x300F: S
x3010: T
x3011: NULL
x3012: x0000, Z
x3013: x0000, K
x3014: x0000, Y
x3015: x0000, Z

```
        .ORIG   x3000
        AND     R0, R0, #0
        LD      R6, EMPTY
        ADD     R1, R0, #6
        JSR     Push                    ;first push
        ADD     R1, R0, #3              ;second push
        ADD     R1, R0, #7
        JSR     Push                    ;third push
        ADD     R1, R0, #2
        JSR     Pop                     ;first pop
        ADD     R1, R0, #1
        JSR     Push                    ;fourth push
        ADD     R1, R0, #13
        JSR     Push                    ;fifth push
        JSR     Pop                     ;second pop
        ADD     R0, R1, #0
        BRnzp   EXIT
Push    STR     R1, R6, #0
        ADD     R6, R6, #-1
        RET
Pop     ADD     R6, R6, #1
        LDR     R1, R6, #0
        RET
EMPTY   .FILL x4000
EXIT    TRAP x25
        .END
```

What is the value at memory location x3FFE after the execution of the third JSR Push instruction?
R0: x0000
R1: x0007
R6: x3FFD

x4000: x0006
x3FFF: x0003
x3FFE: x0007 After the third JSR push, x3FFE = x0007

What is the value in register R1 after the execution of the second JSR Pop instruction?
R0: x000C
R1: x0001
R6: x3FFD

x4000: x0006
x3FFF: x0003
x3FFE: x0001
x3FFD: x0001 After the second JSP pop, R1= x0001

What is the value in register R6 after the execution of the BRnzp EXIT instruction?
R0: x0001
R1: x0001
R6: x3FFD

x4000: x0006
x3FFF: x0003
x3FFE: x0001
x3FFD: x0001 After the program, R6 = x3FFD

```
        .ORIG x3000
        LEA R1, TEXT
PART1   LDR R0, R1, #0
        BRz PART2
        OUT
        ADD R1, R1, #1
        BRnzp PART1

PART2   LEA R1, TEXT
        AND R3, R3, #0
LOOP    LDR R0, R1, #0
        BRz EXIT
        JSR PRINT
        ADD R1, R1, #1
        BRnzp LOOP

EXIT    TRAP x25

PRINT   ADD R3, R3, #1
        ADD R0, R0, R3
POLL    LDI R2, DSR
        BRzp POLL
        STI R0, DDR
        RET

DSR     .FILL xFE04
DDR     .FILL xFE06
TEXT    .STRINGZ "123"
        .END
```

What would the output of this program be?
R1: 1
R0: 2
R2: xFE04
R3: 0001

OUTPUT: 123
xFE06: 2

What will be the value in Register R7 after executing the instruction JSR PRINT?
x300B

MCQ 8

7FF1	k	-2
7FF2	j	-1
7FF3	i	0
7FF4	dyn link	1
7FF5	ret address	2
7FF6	ret val	3
7FF7	a	4
7FF8	b	5
7FF9	c	6
7FFA	d	7
7FFB	z	-2
7FFC	y	-1
7FFD	x	0
7FFE	dyn link	1
7FFF	ret address	2
8000	ret val	3

Read the program below before answering the questions 2 to 10. "…" denotes the code that is irrelevant to this question.

```
int foo(int a, int b, int c, int d);
int main() {
  int x, y, z;
   …
  z = foo(1, x, 2, y);
}

int foo(int a, int b, int c, int d){
  int i, j, k;
  i = c + 2;
   …
  return j;
}
```

Which one of the following LC-3 instruction sequences will be generated for passing variable x to function foo?

```
LDR  R0, rL, X
ADD  R6, R6, #X
STR  R0, R6, #0
```

```
LDR  R1, rL, X
ADD  R6, R6, #-1
STR  R1, R6, #0
```

```
LDR  R0, rL, X
ADD  R6, R6, #1
STR  R0, R6, #0
```

```
LDR  R0, rG, X
ADD  R6, R6, #-1
STR  R0, R6, #0
```

Which one of the following LC-3 instruction sequences will be generated for passing value 1 to function foo?

```
AND  R0, R0, #0
ADD  R0, R0, #1
ADD  R6, R6, #1
STR  R0, R6, #0
```

```
ADD  R0, R0, #1
ADD  R6, R6, #1
STR  R0, R6, #0
```

```
AND  R0, R0, #1
ADD  R6, R6, #-1
STR  R0, R6, #0
```

```
AND  R0, R0, #0
ADD  R0, R0, #1
ADD  R6, R6, #-1
STR  R0, R6, #0
```

Which one of the following LC-3 instruction sequences will be generated for storing the return address in the activation record of function foo?

```
ADD  R6, R6, #-1
STR  R7, R6, #0
```

```
ADD  R6, R6, #1
STR  R5, R6, #0
```

```
ADD  R6, R6, #-1
STR  R5, R6, #0
```

```
STR  R7, R6, #0
ADD  R6, R6, #-1
```

Which one of the following LC-3 instruction sequences will be generated for moving the value of variable j to the slot reserved for storing the return value in the activation record of function foo?

```
LDR  R0, r6, #J
STR  R0, rL, #3
```

```
LDR  R1, rL, #J
STR  R1, rL, #3
```

```
LDR  R1, r6, #J
STR  R1, rG, #J
```

```
LDR  R0, rG, #J
STR  R0, r6, #3
```

Which one of the following LC-3 instruction sequences will be generated for restoring the stack frame pointer before function foo terminates?

```
ADD  R6, R6, #1
LDR  R5, R6, #0
```

```
LDR  R5, R5, #0
ADD  R6, R6, #-1
```

```
LDR  R5, R6, #0
ADD  R6, R6, #-1
```

```
LDR  R5, R6, #0
ADD  R6, R6, #1
```

Which one of the following LC-3 instruction sequences will be generated for assigning the value returned by function foo to variable z and removing the remaining elements of the activation record of function foo from the runtime stack?

```
LDR  R1, R6, #0
STR  R1, rL, Z
ADD  R6, R6, #5
```

```
LDR  R0, R6, J
STR  R0, rL, Z
ADD  R6, R6, #4
```

```
LDR  R1, rL, J
STR  R1, rL, Z
ADD  R6, R6, #5
```

```
LDR  R0, R6, #0
STR  R0, rL, Z
ADD  R6, R6, #4
```

MCQ 9

Read the program below before answering the next three questions.

```
#include <stdio.h>
int foo(int x, int* y);
int main() {
  int a = 2;
  int b = 3;
  a = foo(a, &b);
}

int foo(int x, int* y) {
  int i;
  i = x + *y;
  *y = x;
  return i;
}
```

x2FF5	int i	0
x2FF6	DYN LINK	1
x2FF7	RET ADDRESS	2
x2FF8	RET VAL	3
x2FF9	int x	4
x2FFA	int* y	5
x2FFB	b=3	-1
x2FFC	a=2	0
x2FFD	DYN LINK	1
x2FFE	RET ADDRESS	2
x3000	RET VAL	3

Which one of the following LC-3 instruction sequences will be generated for passing &b to function foo?

```
LDR  R1, rL, B
ADD  R6, R6, -1
STR  R1, R6, 0

ADD  R1, rL, B
ADD  R6, R6, -1
STR  R1, R6, 0

ADD  R1, rG, B
ADD  R6, R6, 1
STR  R1, R6, 0

ADD  R1, rL, B
ADD  R6, R6, -1
STR  R1, R6, B
```

Which one of the following LC-3 instruction sequences will be generated when converting "i = x + *y" to LC-3 assembly language instructions?

```
ADD  R0, rL, Y      R0 = x2FFB (Y points to x2FFB)
LDR  R0, R0, 0      R0 = Y
LDR  R1, rL, X      R1 = X
ADD  R0, R1, R0     R0 = y+x
STR  R0, rL, I      i = y+x

LDR  R0, rL, Y      R0 = Y
LDR  R0, R0, 0      R0 = mem[Y]
LDR  R1, rL, X      R1 = x
ADD  R0, R1, R0     R1 = x +y
STR  R0, rL, I      i = y+x

ADD  R0, rL, Y      R0 = x2FFB
LDR  R0, R0, Y      R0 = Y
LDR  R1, rL, X      R1 = x
ADD  R0, R1, R0     R0 = x + y
STR  R0, rL, I      i = = x + y

LDR  R0, rL, Y
LDR  R1, rL, X
ADD  R0, R1, R0
STR  R0, rL, I
```

Which one of the following LC-3 instruction sequences will be generated when converting "*y = x" to LC-3 assembly language instructions?

```
LDR  R1, rL, X
ADD  R0, rL, Y
STR  R1, R0, 0

LDR  R1, rL, X
LDR  R0, rL, Y
STR  R1, R0, Y

LDR  R1, rL, X
LDR  R0, rL, Y
STR  R1, R0, 0

LDR  R1, rL, X
ADD  R0, rL, Y
STR  R1, R0, Y
```

Read the program below before answering the next four questions. "…" denotes the code that is irrelevant to this question.

```
int main() {
  int a;
  int b;
  int x[10];
  …
  x[2] = 3;
  b = x[a];
  *(x+1) = 0;
}
```

Assume that (i) rG represents the register pointing to the beginning of the locations where the global variables are stored, (ii) rL denotes the register that records the start of the locations where the local variables are stored, and (iii) the offsets of the variables in the program above are given in the table below. In the table, each capital letter represents an integer.

variable	offset
a	A
b	B
x	X

Which one of the following LC-3 instruction sequences will be generated when converting "x[2] = 3" to LC-3 assembly language instructions?

```
AND  R0, R0, 0
ADD  R0, R0, 3
ADD  R1, rL, X          R1 = x2FF1
STR  R0, R1, 2          x2FF3: 3
```

```
AND  R0, R0, 0
ADD  R0, R0, 3
STR  R0, rL, 2
```

```
AND  R0, R0, 0
ADD  R0, R0, 3
ADD  R1, rL, X
LDR  R1, R1, 0
STR  R0, R1, 2
```

```
AND  R0, R0, 0
ADD  R0, R0, 3
LDR  R1, rL, X          R1 = x[0]
STR  R0, R1, 2          x[0] + 2 = 3
```

x2FF1	x[0]	-11
x2FF2	x[1]	-10
x2FF3	x[2]	-9
x2FF4	x[3]	-8
x2FF5	x[4]	-7
x2FF6	x[5]	-6
x2FF7	x[6]	-5
x2FF8	x[7]	-4
x2FF9	x[8]	-3
x2FFA	x[9]	-2
x2FFB	b	-1
x2FFC	a	0
x2FFD	dyn link	1
x2FFE	ret address	2
x3000	ret value	3

Which one of the following LC-3 instruction sequences will be generated when converting "b = x[a]" to LC-3 assembly language instructions?

```
ADD  R1, rL, X          R1 = x2FF1
LDR  R1, R1, A          x2FF1: x[a]
STR  R1, rL, B          b: x[a]
```

```
LDR  R0, rL, A          R0 = a
ADD  R1, R0, X          R1 =
LDR  R1, R1, 0
STR  R1, rL, B
```

```
LDR  R0, rL, X          R0 = x[0]
ADD  R1, rL, A          R1 = x2FFC
LDR  R1, R1, 0          x2FFC: a
STR  R1, rL, B          B = x2FFC
```

```
LDR  R0, rL, A          R0 = a
ADD  R1, rL, X          R1 = x2FF1
ADD  R1, R1, R0         R1 = x2FF1
LDR  R1, R1, 0
STR  R1, rL, B
```

Which one of the following LC-3 instruction sequences will be generated when converting "*(x+1) = 0" to LC-3 assembly language instructions?

```
AND  R0, R0, 0          R0 = 0000
ADD  R1, rL, X          R1 = 2FF1
ADD  R1, R1, 1          R1 = 2FF2
STR  R0, R1, 0          x[1] = 0
```

```
AND  R0, R0, 0          R0 = 0000
ADD  R1, rL, X          R1 = 2FF1
ADD  R1, R1, 1          R1 = 2FF2
LDR  R1, R1, 0          2FF2 = x[1]
STR  R0, R1, 0          x[1] = 0
```

```
AND  R0, R0, 0          R0 = 0
LDR  R1, rL, X          R1 = x[0]
ADD  R1, R1, 1          R1 = x[0] + 1
STR  R0, R1, 0          x[0] + 1 = 0
```

```
AND  R0, R0, 0          R0 = 0
LDR  R1, rL, X          R1 = x[0]
ADD  R1, R1, 1          R1 = x[0] + 1
LDR  R1, R1, 0          R1 = x[0] + 1
STR  R0, R1, 0
```

Read the program below before answering the next three questions.

```
int foo(int a, int* b);
int main() {
  int x=2;
  int y=3;
  int z=4;
  z = foo(x, &y);
}
int foo(int a, int* b) {
  int m[3], n;
  n = a;
  m[0] = b;
  m[1] = *b;
  m[2] = &a;
  a = 12;
  b = 13;
  return *(m+1);
}
```

Which one of the following statements is correct?

After statement "m[1] = *b;" in function foo is executed, the value stored in location 0xAFF2 is 0xAFF9.

After statement "m[1] = *b;" in function foo is executed, the value stored in location 0xAFF2 is 3.

After statement "m[1] = *b;" in function foo is executed, the value stored in location 0xAFF3 is 3.

After statement "m[1] = *b;" in function foo is executed, the value stored in location 0xAFF2 is 0xAFFD.

Which one of the following statements is correct?

After statement "m[2] = &a;" in function foo is executed, the value stored in location 0xAFF4 is 0xAFFC.

After statement "m[2] = &a;" in function foo is executed, the value stored in location 0xAFF1 is 0xAFF9.

After statement "m[2] = &a;" in function foo is executed, the value stored in location 0xAFF2 is 0xAFFC.

After statement "m[2] = &a;" in function foo is executed, the value stored in location 0xAFF4 is 0xAFF8.

xAFF1	n=a
xAFF2	m[0]
xAFF3	m[1]
xAFF4	m[2]
xAFF5	dyn link
xAFF6	ret address
xAFF7	ret value
xAFF8	a
xAFF9	*b
xAFFA	z=4
xAFFB	y=3
xAFFC	x=2
xAFFD	dyn link
xAFFE	ret address
xAFFF	ret value

MCQ 10

If a cache line consists of 32 bytes and we access the byte at memory address 0x4567, what is the range of the addresses of the bytes that are loaded into the cache? Assume only one line of the cache is loaded at a time.

Offset = (2^n = 32 bits for offset) = 5

So, x4560 = 0100 0101 0110 0000 has offset 0
So, x457F = 0100 0101 0111 1111 has offset 31

Range = x4560 to x457F

A direct mapped cache has 8 cache lines. Each cache line consists of 2 words, and each word is one byte. The address bus consists of 7 bits.

8 lines so 8 = 2^3, we have 3 bits for the index. For 2 words holding one byte each, we have 2 = 2^1, so 1 bit for the offset. Hence the tag = 7 - (3+1) = 3

The tag field of the cache consists of 2 bits.

The tag field of the cache consists of 3 bits.

The index field consists of 2 bits.

The index field consists of 1 bit.

A direct mapped cache has 8 cache lines. Each cache line consists of 2 words, and each word is one byte. The address bus consists of 7 bits. Which one of the following statements is correct?

The total number of bytes for storing data in the cache is 8K bytes.

The total number of bytes for storing data in the cache is 8 bytes.

The total number of bytes for storing data in the cache is 16 bytes.

The total number of bytes for storing data in the cache is 32 bytes.

Index = 3 bits
offset = 1 bits
tag = 3 bits

There are 3 cache lines and each line has 1 bytes, there are 2^4bytes in the cache.
= 16 bytes

Read the description below before answering the next four questions.

A direct mapped cache has 8 cache lines. Each cache line consists of 2 words, and each word is one byte. The address bus consists of 7 bits. Assume the cache is empty to start with. Work out if the following accesses to the given addresses are hits or misses. Each access is numbered with a sequence ID for your convenience. All addresses are hexadecimal numbers.

Sequence ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Address	16	20	19	18	11	21	3E	17	11	28	16	29	10	17	21	16

INDEX	TAG	WORD[0]	WORD[1]
0 (000)	010	20	21
1 (001)			
2 (010)			
3 (011)	001	16	17
4 (100)	010	39	29
5 (101)			
6 (110)			
7 (111)	011	3E	3F

TAG = 3 bits
INDEX = 3 bits
OFFSET = 1 bit

ACCESS 1,2,3,4
16 = 0001 011**0** **(MISS)**
20 = 0010 000**0** **(MISS)**
19 = 0001 1001 **(MISS)**
18 = 0001 1000 **(HIT)**

ACCESS 5, 6, 7
11 = 0001 0001 (MISS)
21 = 0010 0001 (MISS)
3E = 0011 1110 (MISS)

ACCESS 8, 9, 10, 11, 12
17 = 0001 0111 (HIT)
11 = 0001 0001 (MIS)
16 = 0001 0110 (HIT)
29 = 0010 1001 (MISS)

ACCESS 13, 14, 15
10 = 0001 0000 = (HIT)
17 = 0001 0111 = (HIT)
21 = 0010 0001 = (MISS)