Breakout session

1. For the following assume that values A, B, and C reside in memory. Also assume that instruction operation codes are represented in 8 bits, memory addresses are 64 bits, and register addresses are 6 bits.

a. For each instruction set architecture shown below, how many addresses appear in each instruction for the code to compute **C=A+B**, and what is the total code size?

Register Architecture is done for you as an example:

Register	No. of addresses or names	Total code size
Load R1,A	2	8+64+6=78
Add R3,R1,B	3	8+64+6+6=84
Store R3, C	2	8+64+6=78

Solution:

STACK	No. of addresses or names	Total code size

Accumulator	No. of addresses or names	Total code size

Load-Store	No. of addresses or names	Total code size

2. The value represented by the hex number 434F4D5055544552 is to be stored in an aligned 64-bit double word.

a. Using the physical arrangement of the first row in Figure A.5, write the value to be stored using Big Endian byte order.
Next, interpret each byte as an ASCII character and below each byte write the corresponding character, forming the character string as it would be stored in Big Endian order.

b. Repeat a using Little Endian byte order.

Solution:

a-

Big Endian:

Value							
0	1	2	3	4	5	6	7

b-

Little Endian:

Value							
0	1	2	3	4	5	6	7

3. Your task is to compare the memory efficiency of four different styles of ISAs: (Accumulator, stack, load-store, and memory-memory).

To measure memory efficiency, make the following assumptions:
- All instructions are an integral number of bytes in length;
- The opcode is always 1 byte;
- Memory access use direct, or absolute addressing;
- The variables A, B, C, D are initially in memory.

a. Invent your own assembly language mnemonics (Figure A.2 provides a useful sample to generalize), and for each architecture write an (does not have to be the optimized) equivalent assembly language code for this high-level code sequence:

A=B+C;
B=A+C;
D=A-B

Solution:

A=B+C
B=A+C
D=A-B

a-

Accumulator: Your proposed Program

Stack: Your proposed Program

Load-Store: Your proposed Program

Memory-Memory: Your proposed Program

4. What is the pipelining timing diagram FDXMW for the following program using no forwarding unit and using forwarding unit?

1)

Add R1, R2, R3

BNEZ R1, LL

2)

SW R0, 0 (R1)

Add R2, R3, R1