1 (1 mark). “All of the signed integer values between -16 and 16 inclusive can be represented as 4 bit 2’s complement integers”

False

Explanation: There are 33 signed integer values between -16 and 16, but only 16 distinct values can be represented using 4 bits

2 (1 mark). “Loop interchange can improve cache performance of the C code

```
float X[1024][1024], Scale;
int a, b, c;
for (c = 0 ; c <10; c++)
    for (b = 0; b < 1024; b++);
        for (a = 0; a < 1024; a++)
            X[b][a] = X[b][a] * Scale;
```

Interchanging a-loop and b-loop doesn’t help as they generate accesses in storage order exploiting spatial locality of reference

However, temporal locality of reference can be improved by interchange c-loop/b-loop followed by interchange c-loop/a-loop

4 a (0.5 mark). Explain: Maximum Vector Length

Maximum vector size that can be operated on by a vector instruction of a given processor

4 b (0.5 mark). Explain: Stripmining

Technique to handle a vectorizable loop whose iteration count is higher than the maximum vector length of the target processor

3. Given the 2-dimensional C array float Y [512] [511]; with base address 0xA0000000

(a) (0.5 mark) What is the address of Y [8] [0]?

Array element size = sizeof(float) = 4 Bytes

Address(Y[8][0]) = Address(Y[0][0]) + 8 * Row size

= 0xA0000000 + 8 * 511 * 4

(b) (0.5 mark) If the program is run on a computer with data cache block size of 64 Bytes, which array elements are in the same block as Y [0] [511]?

There is no array element Y [0] [511] as the column dimension of Y is 511 and indices are therefore from 0 to 510

5 (1 mark). Would you expect the code below to be auto-vectorized? If not, transform it to facilitate auto-vectorization.

```
float Z[1024][1024], Sum = 0.0; int a,b;
for (a=0; a<1024; a++)
    for (b=0; b<1024; b++)
        Sum = Sum + Z[a][b];
```

No, the inner b-loop will not be vectorized due to the dependence introduced by the scalar variable Sum

```
float rowSum[1024]; /*initialize to 0 */
for (a=0; a<1024; a++)
    for (b=0; b<1024; b++)
        rowSum[b] = rowSum[b] + Z[a][b];
int i; /* i-loop below is not vectorizable */
for (i=0; i<1024; i++) Sum = Sum + rowSum[i];
```