CSU34021 Tutorial 1 Notes

(i)
```
mov     eax, [epb+12]
push    eax
```

can be simplified by using

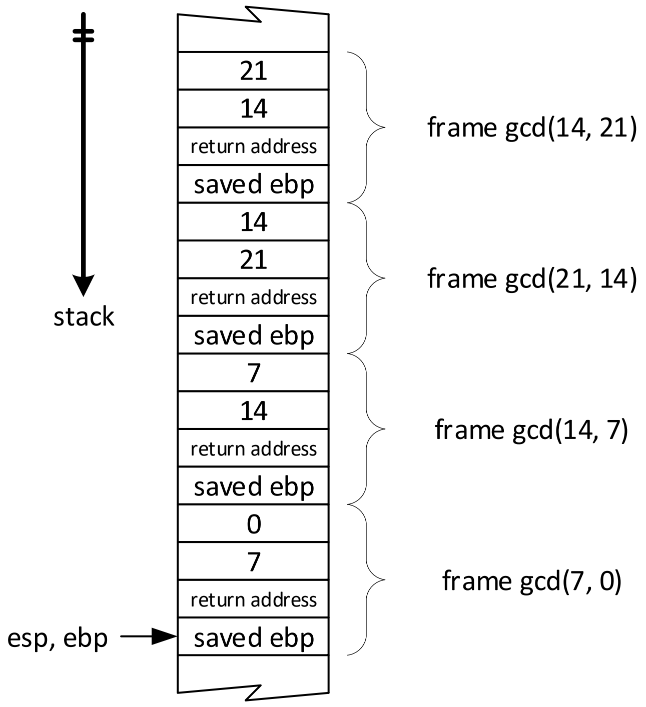
```
push    [ebp+12]
```

(ii) Some students pushed the parameters (particularly to p) in the wrong order. Although function p would return the correct result, it is incorrect coding.

(iii) a%b should be calculated using idiv. idiv uses signed arithmetic whilst div uses unsigned arithmetic. idiv divides edx:eax (64 bits) by the instruction operand (32 bits). The quotient is returned in eax and the remainder in edx. edx should be initialised using cdq as it sign extends eax across edx. Zeroing edx is not the same, although it will work with the examples given (need better test cases to catch this error)

(iv) Some students had trouble with global variable g which needs to be allocated in t1.asm and its "interface" specified in t1.h (see sample answer).

(v) Layout of stack frames must match submitted code. Some students thought that each function had its own stack which is not the case.



4 stack frames – 4 x 32 bit DWORDs per frame

(v) Some students had difficulties using a development environment. Computer Science / Engineering students need to be comfortable using environments such as Visual Studio and Eclipse. They all tend to be very similar.

Additional comments from Harshvardhan Pandit:

(vi) Some students were confused with the depth of the stack frames in Q2. The total number of stack frames should equal the total number of recursive calls made, as each call will result in storing of parameters in its own stack frame.

(iv) Additionally, some students also had a confusion with the calculation of what constitutes a stack frame – in this case, 1 stack frame is the collective set of all memory values stored on the stack to save the state for a call. So, in the above diagram example, 1 stack frame has four parameters/data values. The confusion was based on considering each data value as its own stack frame.