Section 1

Question a

i) Advantages:

* We can perform normal addition and subtraction on 2C numbers to achieve meaningful results
* There is only one representation of 0

Disadvantages:

* The range of numbers is not symmetrical
* Conversion is more complicated and may require additional hardware in CPU

ii) Advantages:

* Simple to both understand and convert
* Symmetrical

Disadvantages:

* Can’t perform addition or subtraction directly
* Storing two different kinds of information in the same value

Question b

i) -218

ii) 10110 11010

iii) 00001 00000

iv) -332

Question c

i) Binary Coded Decimal

ii) 0010 0010 0001 1101

Question d

i) LOADA Addr: R = MEM[Addr]

STORE Addr: MEM[Addr] = R

LOADC Const: R = Const

ADD Addr: R = R + MEM[Addr]

SUB Addr: R = R - MEM[Addr]

BRZ Addr: PC = Addr (R < 0)

ii) UB

0x0 (value 1)

0x1 (value 2)\

0x2 (result)

.

.

0x100 LOADA 0x0 ## Load Register with M[0]

0x101 ADD 0x1 ## R = R + M[0001]

0x102 STORE 0x2 ## M[0002] = R

0x103 LOADC 9 ## R = 9

0x104 S 0x2 ## R = 9 – M[0002]

0x105 BRZ 0x107 ## if R < 0 then jump to LoadC i.e M[0002] > 2

0x106 STOP

0x107 LOADC 6 ## R = 6 = 0110

0x108 ADD 0x2

0x109 STORE 0x2

0x10A STOP

Section 2

NB: I’m not sure how much of this past paper pertains to this years exam (Jana’s content)

Question apushpush

(Not sure)

Quad word (8 bytes), long (4 bytes), word (2 bytes) and byte. We can refer to parts of a register which are exactly of this type by the name of the register e.g %al/%ah is a byte in size, %ax is a word, %eax is long and %rax is quad word in size.

Question b

C1D0 0000 = 11000001110100000000000000000000 or -26

C1DE 0000 = 11000001110111100000000000000000 or -27.75

Question c

4300 0000 - 01000011000000000000000000000000

C301 0000 - 1100001100001 0-

BF80 0000

Question d

rec:

pushq %rbx ; make space for local variable c

pushq %rbp ; make space for local variable d

movq $0x1, %rbx ; c = 1 -- not sure tho

movq %rdi, %rbp ; d = a

subq %rsi, %rbp ; d = d - b (so d = a - b)

cmpq $0, %rbp ; compare d and 0

jle .else ; jump to else if d <= 0

pushq %rdi ; swap %rdi and %rsi around with help of stack

movq %rsi, %rdi

popq %rsi

call rec ; rec(b,a)

.else:

.test:

cmp %rsi, %rdi

jg .endloop ; if a > b jump to endloop

addq %rbx, %rdi ; a = a + c

salq $0x1, %rbx ; c = 2 \* c <=> c << 1

jmp .test

.endloop:

movq %rbx, %rax ; set return value

popq %rbp

popq %rbx

ret