2018 – 2019

1. (a)(i) (A’ ∙ C’)’ ∙ (A ∙ D + A ∙ D’) + A ∙ C + C

= ((A + C)’)’ ∙ (A ∙ D + A ∙ D’) + A ∙ C + C De Morgan’s

= (A + C) ∙ (A ∙ D + A ∙ D’) + A ∙ C + C Negation

= (A + C) ∙ (A ∙ (D + D’)) + A ∙ C + C Distributive

= (A + C) ∙ (A ∙ 1) + A ∙ C + C Negation

= (A + C) ∙ A + A ∙ C + C Simplification

= (A + C) ∙ A + A ∙ C + C ∙ 1 Simplification

= (A + C) ∙ A + C ∙ (1 + A) Distributive

= (A + C) ∙ A + C ∙ 1 Simplification

= (A + C) ∙ A + C Simplification

= A ∙ A + A ∙ C + C Distributive

= A + A ∙ C + C Idempotence

= A + A ∙ C + C ∙ 1 Simplification

= A + C ∙ (A + 1) Distributive

= A + C ∙ 1 Simplification

= A + C Simplification

(b) See lectures.

(c) The Big Endian format stores the most significant byte first while the Little Endian format stores the least significant byte first.

Assuming the integer is a 32-bit integer: 32-bit memory requires padding:

Big Endian: 00 00 87 07

Little Endian: 07 87 00 00

Otherwise, assuming that memory is byte-addressable:

Big Endian: 87 07 or 1000 0111 0000 0111

Little Endian: 07 87 or 0000 0111 1000 0111

(d)(i) 4G × 64bits = 4G × 8bytes = 32GB

(ii) 4G / 1024M = 4G / 1G = 4

(iii) Number of chips in one module: 64bits / 8bits = 8 = 23

Number of bytes in one chip: 1024M × 8 bits = 210 × 220 × 1 byte = 230 bytes

30 + 3 = 33 bits are needed

(iv) 29 / 1024 = 0 (R 29)

Memory module 0

(v) 29 mod 4 = 1

Memory module 1

(e) 0x426A0000 à 0100 0010 0110 1010 0000 0000 0000 0000

Sign: 0

Exponent: 1000 0100

Significand: 1101 0100 0000 0000 0000 000

0x41D00000 à 0100 0001 1101 0000 0000 0000 0000 0000

Sign: 0

Exponent: 1000 0011

Significand:

Since the exponent for 0x41D00000 is smaller, the significand needs to shift one bit to the left à 1101 0000 0000 0000 0000 000

01.1101 0100 0000 0000 0000 000

+ 00.1101 0000 0000 0000 0000 000

10.1010 0100 0000 0000 0000 000

Therefore, the exponent needs to be increased by one bit: 1000 0101

The new significand is now: 0101 0010 0000 0000 0000 000

The binary sum is 0 10000101 010 1001 0000 0000 0000 0000

The IEEE sum is 0x42A90000

1. (a)(i) see lectures. Eip. Jump instructions or goto?

(ii) see lectures. Programmed, interrupt, DMA, processor. Cycle stealing

(iii) see lectures. Push input params onto stack from R to L. Call method (implicit push eip return addr onto stack). Remove params from stack (add esp) and call return (implicitly returns to return addr).

(iv)

* Function call has no context change
* Other interrupts are disabled during interrupt
* Function call is synchronous, interrupt is asynchronous

(b)

Addr Assembler Instr. Comments

0 0 A

1 X X

2 Y Y

3 3 3

4 Z Z

5 1F0H Stack memory

…

10H PUSH [1]

11H PUSH [2]

12H MULT X × Y

13H PUSH [3]

14H PUSH [4]

15H MULT 3 × Z

16H SUB X × Y – 3 × Z

17H PUSH [1]

18H PUSH [2]

19H ADD X + Y

20H MULT (X × Y – 3 × Z) × (X + Y)

21H POP [0] A = (X × Y – 3 × Z) × (X + Y)

22H STOP

Should end with store A, a should be in a memory location e.g. 6, store[6H] // A