Question 1

b

- When we store the control words with the next-state information in a control memory. We then use the control inputs and status signals to select the appropriate address of the next state.
- Since we're not storing control words, they cannot dynamically depend on the values of control input. Additional states are supplied instead.
- An init state and load state would be added.
- This means we have five states, so a 3-bit Control Address Register is introduced.
- A multiplexer is accessed to access the control input, where the result determines which of the two stored addresses is used.

Question 2

a

Memory

This contains sequences of instructions, which together forms the program. It accessed the instruction at the memory address that it loads in. These instructions are in the format:

Opcode	DA	SA	SB
7-bit	3-bit	3-bit	3-bit

Control Memory

Each instruction can take several clock cycles to perform. An instruction loads the corresponding microprogra from control memory. Each microprogram consists of one or more control words which controls the state of the processor for that clock cycle.

Registers

Program Counter

A program counter contains the address in memory of the instruction that is currently being performed. The program counter has two control signals -

Program Load and Program Increment.

State	PL	PI
Unchanged	0	0
Incremented	0	1
Offset	1	0

Instruction Register

This contains the instruction that is currently being performed. It extracts the opcode from the instruction and passed it into the CAR. It has the Instruction Load signal.

State	IL
Unchanged	0
Load from Memory	1

Control Address Register

This contains the address of the current control word being performed in control memory. The CAR changes every clock cycle.

State	Mux S
Incremented Mux C	0
Loaded from Mux C	1

Processor Operation

Fetch

The microprogram for the fetch next instruction from memory consists of 3 control words which: 1. Increment the PC and load the instruction from memory (PL=0, PL=1) 2. Write instruction to IR (IL=1) 3. Write opcode to CAR (Mux S=1, ie. MS=0001) On the next clock cycle the Control Memory will access the address of the opcode stored in the CAR. At the end of each microprogram, the next instruction will need to be fetched. Therefore it makes sense for the location of the fetch microprogram to be stored in the next address field of the last control word in a microprogram.

Branching

In the case of a branch, the PC would need to load the next instruction from the memory location specified in the branch instruction. In this case, PL is set and the location of the instruction is the concatenation of DR and SB, stored in the Instruction Register.

In a conditional branch, MS is set to correspond to the flag to be checked. If the flag is 0 (false) then the CAR increments the address that is in Mux C which would be the fetch next instruction. Otherwise, the address is loaded from Mux C which will be the unconditional branch microprogram, followed by the fetch next instruction microprogram.

Address	Operation
0x60	Conditional Branch
0x61	Unconditional Branch
0x62	Fetch

b

1. Fetch next instruction
    1. Increment PC and load from memory
    2. Write instruction to IR
    3. Write opcode to CAR
2. Access control word of ADI
3. Set control signals of microprogram
    - MB=1 (immediate value)
    - RW=1
    - FS=00010
    - MD=0
    - MC=0
    - MS=0
    - NA=next control word address
    - PL=0
    - PI=0
    - IL=0
    - MM=X

Question 3

b

- The boolean expression implements a carry look-ahead adder.

- For a ripple adder, we'd calculate the carry of each stage by looking at the carry out of the previous stage. This creates a large propagation delay.
- Using the carry look-ahead adder, the expected carry-in is calculated using the equation. The trade off is more redundant gates, but with a descreased propagation delay.
- As the number of bits increases, the size of gates increases, which causes an increase in power usage, size, delay, etc. so the efficiency decreases the large the adder is.