Computer Organization and Architecture

Chapter: Control Unit Design Issue

Exercise & Exam-Oriented Questions with Answers — Set 2

This is a second, fresh set of short-answer and broad (long-answer) questions from the chapter Control Unit Design Issue, covering additional exam-relevant concepts not repeated from Set 1 — including address sequencing, mapping and branching logic, PLA-based control, nanoprogramming, and encoding techniques. Use this set alongside Set 1 for complete revision.

Section A — Short Answer Questions

Q. What is meant by "address sequencing" in a microprogrammed control unit?

Ans. Address sequencing refers to the technique used to determine the address of the next microinstruction to be executed. It involves selecting between the next-address field of the current microinstruction, a mapped address derived from the instruction opcode, or a branch address based on a status condition.

Q. What is the mapping process in a microprogrammed control unit?

Ans. Mapping is the process of converting the opcode of a machine instruction into the starting address of the corresponding microroutine in control memory, usually done using fixed mapping logic or a small lookup table.

Q. What is a branch microinstruction?

Ans. A branch microinstruction is a microinstruction that can alter the normal sequential execution of a microprogram by transferring control to a different address in control memory, depending on the value of a specified status or condition flag.

Q. What is meant by "conditional branching" in control memory addressing?

Ans. Conditional branching means that the address of the next microinstruction depends on the value of a selected status bit (such as zero, carry, or a mode bit). If the condition is true, the sequencer branches to a specified address; otherwise, it continues with the next sequential address.

Q. What is a Programmable Logic Array (PLA), and where is it used in control unit design?

Ans. A PLA is a programmable digital logic device that can implement any combinational logic function using an AND array followed by an OR array. It is often used to implement the fixed logic of a hardwired control unit in a more structured and compact way than discrete gates.

Q. What is nanoprogramming?

Ans. Nanoprogramming is a two-level microprogramming technique in which a small nanomemory stores the actual encoded control words (nanoinstructions), while a larger microprogram memory stores pointers to the nanomemory. It reduces control store size for highly repetitive control patterns.

Q. What is meant by "residual control"?

Ans. Residual control refers to control functions that are handled through conventional hardwired logic even within an overall microprogrammed control unit, typically used for time-critical or infrequently changing operations, combining the two approaches.

Q. What is the role of the sequencing logic (next-address generator) in a microprogrammed control unit?

Ans. The sequencing logic determines the address of the next microinstruction to be fetched, based on the next-address field of the current microinstruction, the mapped address from the opcode, or a branch address selected according to status flags.

Q. Differentiate between a microoperation and a microinstruction.

Ans. A microoperation is a single elementary operation performed on data stored in registers, such as a shift, add, or transfer, during one clock cycle. A microinstruction is the control word that specifies one or more such microoperations to be carried out together in that clock cycle.

Q. What is meant by "encoding" in the context of vertical microprogramming?

Ans. Encoding refers to grouping related, mutually exclusive control signals into a compact binary field within the microinstruction, rather than assigning a separate bit to each signal. A decoder then expands this encoded field back into individual control signals.

Q. Why does encoding reduce the width of a microinstruction but not the amount of control information conveyed?

Ans. Encoding groups mutually exclusive signals (where only one can be active at a time) into a smaller field using binary combinations, so fewer bits are needed to represent the same number of possible signals; the decoder later expands this field to activate the correct individual control line.

Q. What is meant by the "design complexity" trade-off between horizontal and vertical microprogramming?

Ans. Horizontal microprogramming needs a wider control word and larger control memory but simpler decoding logic, while vertical microprogramming needs a narrower control word and smaller control memory but more complex decoding circuitry — so reducing word width increases decoding complexity, and vice versa.

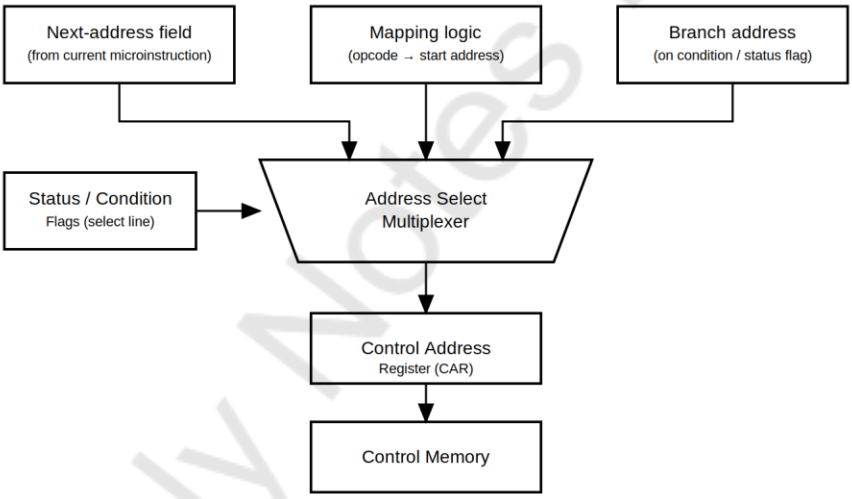
Section B — Broad / Long Answer Questions

Q. Explain, with a suitable diagram, how the address of the next microinstruction is determined in a microprogrammed control unit.

Ans. In a microprogrammed control unit, the address of the next microinstruction is not always simply the next sequential address; it must be determined using address sequencing logic that can choose from three possible sources:

- The next-address field present within the current microinstruction, when execution is to continue sequentially or to a specified fixed address.
- The mapped address, generated by mapping logic that converts the opcode of the machine instruction into the starting address of its microroutine, used when a new machine instruction begins execution.
- The branch address, selected when a conditional branch microinstruction is executed and a specified status flag (zero, carry, mode bit, etc.) determines whether to branch.

These three possible addresses are fed into an address-select multiplexer, which uses the relevant status/condition flag as its select input to choose the correct address. The selected address is loaded into the Control Address Register (CAR), which is then used to access the next microinstruction from control memory.



Address sequencing / branching logic in a microprogrammed control unit

This mechanism allows the control unit to support looping, conditional execution, and jumping between microroutines, which is essential for implementing machine instructions that involve conditional behaviour, such as conditional jumps or multi-cycle operations.

Q. Explain the concept of mapping and branching in microprogram control memory with an example.

Ans. Mapping is the technique of translating the opcode of a machine-level instruction into the starting address of the corresponding microroutine stored in control memory, so that when a new instruction is fetched, the control unit can directly locate where its microprogram begins, instead of searching sequentially.

For example, if an instruction has an opcode of 5 bits, a simple mapping scheme may place two fixed zero bits in front of the opcode and pad the remaining bits with zeros, generating a unique starting address in control memory for each distinct opcode. This is commonly implemented using a small mapping ROM or a fixed logic circuit.

Branching, on the other hand, occurs within a microroutine, where a branch microinstruction tests a status/condition flag (such as a carry or zero flag) and directs the sequencer to jump to a different address in control memory if the condition is satisfied, or to continue sequentially otherwise. Together, mapping (opcode to microroutine) and branching (within a microroutine) allow a microprogrammed control unit to correctly execute the many different instructions and conditional paths of an instruction set using a single shared control memory.

Q. What is a Programmable Logic Array (PLA)? Explain its role in implementing a hardwired control unit.

Ans. A Programmable Logic Array (PLA) is a general-purpose combinational logic device consisting of a programmable AND array followed by a programmable OR array, which can be configured to implement any set of Boolean (sum-of-products) functions.

In a hardwired control unit, instead of designing control logic using separate, individually wired gates and flip-flops for every control signal, a PLA can be used to implement the entire combinational control logic in a compact, structured, and more easily verifiable form. The inputs to the PLA are the decoded opcode bits, timing signals, and status flags, and its outputs are the required control signals. Because a PLA follows a regular, programmable structure, it is easier to design, lay out, and test compared to random (ad-hoc) gate-level wiring, though it still shares the fundamental limitation of hardwired design — any change to the control logic requires reprogramming or redesigning the PLA array, not simply editing stored data as in a microprogrammed design.

Q. What is nanoprogramming? Explain how it differs from ordinary (single-level) microprogramming.

Ans. Nanoprogramming is a two-level implementation of microprogrammed control, introduced to reduce the size of the control memory needed to store highly repetitive control-word patterns.

In ordinary (single-level) microprogramming, each microinstruction directly contains the encoded or unencoded control-word bits needed to generate control signals, and every distinct microinstruction requires its own full-width word in control memory. In nanoprogramming, a small memory called the nanomemory stores the actual distinct control words (nanoinstructions), while a separate, larger microprogram memory stores short pointers (nanoaddresses) to entries in the nanomemory rather than storing the full control words directly.

Since many microinstructions across different microroutines often need the same or similar control-word patterns, storing each unique pattern only once in the nanomemory, and referencing it by a short pointer from the microprogram memory, considerably reduces the total memory required, though at the cost of an additional level of addressing and access delay.

Q. Explain the concept of residual control and how it can be combined with microprogrammed control.

Ans. Residual control refers to the practice of handling certain control functions using conventional hardwired (fixed) logic, even in a system whose overall control unit is primarily microprogrammed.

This hybrid approach is used because a few operations — particularly those that are extremely time-critical, very simple, or rarely changed — can be implemented more efficiently and with less delay using direct hardwired logic than by fetching a microinstruction from control memory. The bulk of the control unit's functionality, especially the complex and frequently modified parts, is still handled through microprogramming for flexibility, while the small residual portion uses hardwired logic for speed. This combination allows designers to balance the flexibility of microprogramming with the speed advantages of hardwired control for the most critical control paths.

Q. Explain encoding and decoding of control signals in vertical microprogramming with the trade-offs involved.

Ans. In vertical microprogramming, related and mutually exclusive control signals (signals of which only one can be active in a given microoperation) are grouped together and represented using a small encoded binary field, instead of assigning one dedicated bit to every individual signal as in horizontal microprogramming.

For example, if a microinstruction needs to select one of eight possible ALU operations, a horizontal format would require eight separate bits (one per operation, only one of which is ever 1), whereas a vertical (encoded) format can represent the same eight operations using only 3 bits, since $2^3 = 8$ unique binary combinations are enough. A decoder is then required after the control memory to expand this 3-bit field back into the correct one-of-eight control signal.

The trade-off is that encoding significantly reduces the microinstruction word length and hence the size of control memory required, but it introduces decoding delay (since a decoder circuit must interpret the encoded field before the actual control signal becomes available) and reduces the degree of parallelism, since encoded fields typically allow only one operation from each group to be selected per microinstruction, rather than allowing many independent signals to be active simultaneously as in the horizontal format.