Chapter 05: Basic Processing Units … Control Unit Design

Lesson 15: Microinstructions

Objective

- Understand that an instruction implement by sequences of control signals generated by microinstructions in microprogram concept
- Learn how the microinstructions store a sequence of micro-operations that are used to implement the instruction
- Learn how the three fields at each microinstruction, function select, data route select and storing input-output unit select field

Microprogram concept

Microprogram concept

- Hardware does not directly execute an instruction without a large number of logic gates
- Instead, the hardware executes very simple micro-operations defined by microinstructions stored at control memory
- Each instruction specifies a sequence of micro-instructions that are used to implement the instruction

Microprogram concept

- Each instruction translated into a short program of microinstructions by the hardware
- Similar to the way a compiler translates each instruction in a high-level language program into a sequence of assembly-language instructions

Microinstructions for ADD *r1*, *r*2, *r*3 instruction

Micro instruction operation to execute the ADD *r1*, *r*2, *r*3 instruction

- Actions (micro-operations) for the data path and control sequences
- Fetch the instruction from memory address I pointed by PC
- (i) Microinstruction *j*: PC → MAR
- (ii) Microinstruction *j* + 1: PC ← PC + 4 for 32 bits memory word alignments

Fetch instruction operation

(iii) Microinstruction $j + 2$: Activate signal ALE for one cycle.

(iv) Microinstruction $j + 3$: Activate signal MEMRD

(v) Microinstruction $j + 4$: $M[I] \rightarrow MDR$

(vi) Microinstruction $j + 5$: Deactivate signal MEMRD

(vii) Microinstruction $j + 6$: $MDR \rightarrow IR$

Microprogrammed translation of the instruction ADD *r*1, *r*2, *r*3

- Six micro-operations to perform add arithmetic operation:
  1. Microinstruction i: $r2 \rightarrow X$. (read value of $r2$)
  2. Microinstruction $i + 1$: $X \rightarrow$ ALU (send it one input of the adder)
  3. Microinstruction $i + 2$: $r3 \rightarrow Y$ (reads the value of $r3$)

Micro-operations to perform add arithmetic operation

4. Microinstruction i + 3: $Y \rightarrow$ ALU input of the adder
5. Microinstruction i + 4: Selects through gates j an operation for ADD using ID for instruction received at IR during 'a' microinstructions j to j +6, (Z) $\leftarrow$ ALU (gets the addition result) and transfers status flags generated, carry and overflow to status register. (Status Register) $\leftarrow$ ALU (the carry and overflow)
6. Microinstruction i + 5: r1 $\leftarrow$ Z. (send the addition result in r1)

Microprogram of ADD $r1$, $r2$, $r3$ instruction at control memory addresses a_i to a_{i+5}

Address	Microinstruction Symbolic representation
a_i	$r2 \rightarrow X$
a_{i+1}	$X \rightarrow$ ADDER
a_{i+2}	$r3 \rightarrow Y$
a_{i+3}	$Y \rightarrow$ ADDER
a_{i+4}	$Z \leftarrow$ ADDER, Status C and OV $\leftarrow$ ADDER
a_{i+5}	$r1 \leftarrow Z$

Each micro-operation taking one control clock cycle

- Instruction requires six control clock cycles to complete
- Here, one processor control clock cycle period equals the period of one step after the increment of the Control Sequence address incremental/ Counter The PC after the end of the microprogram points to the next instruction due to the microinstruction $j + 2$
- The processor fetches the next instruction from the memory by the microinstructions j to $j + 6$

Stored Microprogram for generating control signals for ADD $r1$, $r2$, $r3$

- Let a_i, a_{i+1}, .. be the sequence of address outputs for executing the microprogram
- Let $s1$ to $s12$ be the storing unit control signals at successive addresses

Stored Microinstructions

- 1. Storing register *r*1 output control
- 2. Storing register *r*1 input control
- 3. Storing register *r*2 output control
- 4. Storing register *r*2 input control
- 5. Storing register *r*3 output control
- 6. Storing register *r*3 input control

Stored Microinstructions

- 7. Input for arithmetic unit *X* output control
- 8. Input for arithmetic unit *X* input control
- 9. Input for arithmetic unit *Y* output control
- 10. Input for arithmetic unit *Y* input control
- 11. Arithmetic unit output *Z* output control
- 12. *Z* input control

Stored Microinstructions

- Let f1 to *f*2 be the function select control signals
  1. Data transfer
  2. Add

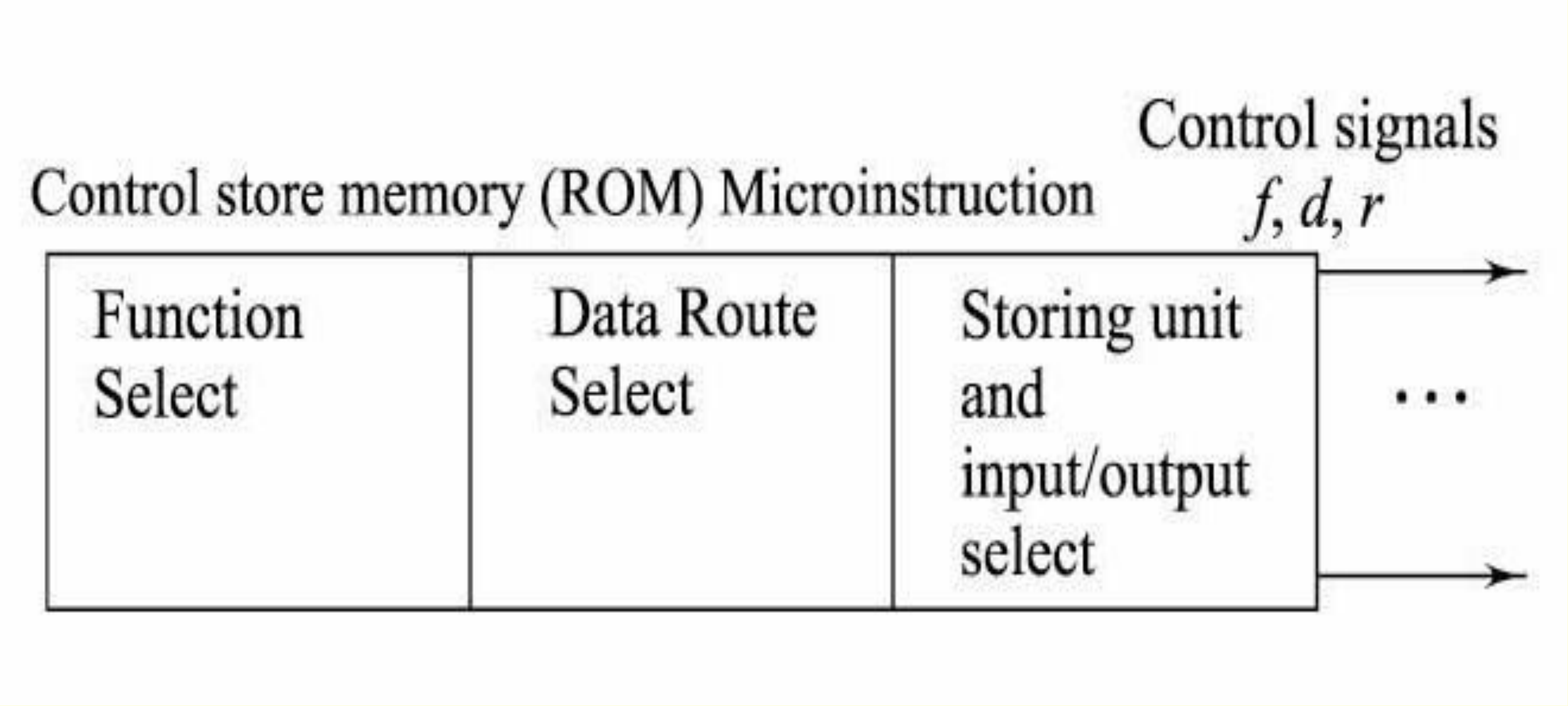
Data route internal bus select control signal

- Let $d1$ be the data route internal bus select control signal
- Let 0 represent inactive, and 1 active

Stored microinstructions at each address for the microprogram for ADD *r1*, *r2*, *r3*

Address	*f*1	*f*2	*d*1	*s*1	*s*2	*s*3	*s*4	*s*5	*s*6	*s*7	*s*8	*s*9	*s*10	*s*11	*s*12
a_i	1	0	1	0	0	1	0	0	0	0	1	0	0	0	0
a_{i+1}	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
a_{i+2}	1	0	1	0	0	0	0	1	0	0	0	0	1	0	0
a_{i+3}	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0
a_{i+4}	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
a_{i+5}	1	0	1	0	1	0	0	0	0	0	0	0	0	1	0

Three fields at a microinstruction word in control memory



Summary

We learnt

- Microinstructions
- Microinstruction at each address has
- Three fields at each microinstruction
- Function select field
- Data route select field
- Storing input-output unit select field

End of Lesson 15 on

Microinstructions