Chapter 05: Basic Processing Units … Control Unit Design Organization

Lesson 03

Register transfers

Objective

- Learn a register transfer
- Register transfer language (representation)

Register transfer

Output

- An output can be in two states: tristate and active
- Certain logic design defines active state = 1 and certain =0
- Tristate means disconnected state in which logic state is neither 0 nor 1

Register transfer or register move or load

- Register transfer or register move─ sending the information bits to a register from other register
- Register load─ means sending the information bits to a register from memory

Register transfer or register move or load

- A microoperation in which a register gives the bits at the output to enable the input of these at another register
- The transfer occurs only if a control input, which permits load of bits the input register active
- The instance at which the transfer occurs is controlled by clock input transition from logic 0 to 1 or 1 to 0

Representing an instruction for register transfer

- MOV r1, r2
- The r2 and r1 are called the operands
- The r1 and r2 are the registers that store the bytes required for the operation
- The registers r1 and r2 are the source and destination operands, respectively

Microoperation for Register transfer

- Consider *n*-bit (8 or 16 or 32) register rj
- rj can save (register the *n*-bits) when a write (≡ load) control signal Cr is applied to it for an interval T
- The register saves at an instance at which transfer occurs and that is controlled by clock input transition from logic 0 to 1 during the interval T

Microoperation for Register transfer from ri to rj

- Assume— a register ri connects to another register rj using *n* interconnections
- The register ri transfers the bits to rj when the load (called control) input is active and a clock pulse is applied to it during T

The microoperation

- Written as Cr: rj ← ri in the register transfer language
- When Cr = 1 (active) the transfer [means bits writing (loading)] takes place at rj from ri

Register Transfer Language

- number of microoperations which are required to execute an instruction by a processing unit. A register transfer language enables proper expressions for these microoperations.
- A character or set of few characters specify a register. For example, MAR specifies memory address register at the processor.
- 2. An arrow shows the direction of transfer. For example, rj ← ri in a register transfer in direction from ri to rj.
- 3. A control input for a select interval is given a name and a colon is placed after it. For example, Cr: rj ← ri.
- 4. A set of microoperations are shown by a comma separating them. For example, rj ← ri., rm ← rn.
- 5. A set of bits on which operations takes place are specified in the parentheses.. For example, rj (15-8) ← ri. (L). This means bit b8 to b15 in rj are transferred from lower byte of ri..

- number of microoperations which are required to execute an instruction by a processing unit. A register transfer language enables proper expressions for these microoperations.
- A character or set of few characters specify a register. For example, MAR specifies memory address register at the processor.
- 2. An arrow shows the direction of transfer. For example, rj ← ri in a register transfer in direction from ri to rj.
- 3. A control input for a select interval is given a name and a colon is placed after it. For example, Cr: rj ← ri.
- 4. A set of microoperations are shown by a comma separating them. For example, rj ← ri., rm ← rn.
- 5. A set of bits on which operations takes place are specified in the parentheses.. For example, rj (15-8) ← ri. (L). This means bit b8 to b15 in rj are transferred from lower byte of ri..

Register transfer language

- Number of microoperations required to execute an instruction by a processing unit
- A register transfer language enables proper expressions for these microoperations

Register specification

1. A character or set of few characters specify a register
- For example, MAR specifies memory address register at the processor

Use of arrow for Register transfer specification

2. An arrow shows the direction of transfer

- For example, rj ← ri in a register transfer in direction from ri to rj

Register transfer language

- r1 ← r2─ The arrow shows the direction (right to left) of transfer from r2 to r1
- r1 → r2─ The arrow shows the direction (right to left) of transfer from r1 to r2

Register transfer language

- Data bus ← r1 means from r1 the data bits are placed at the data bus
- Address bus ← PC means from program counter register the data bits are placed at the data bus

Control input

3. A control input for a select interval is given a name and a colon is placed after it. For example, Cr: rj ← ri

A set of microoperations

5. A set of microoperations shown by a comma separating them
- For example, rj ← ri., rm ← rn

Use of parentheses

6. A set of bits on which operations takes place are specified in the parentheses

- For example, rj (15-8) ← ri. (L). This means bit b8 to b15 in rj are transferred from lower byte of ri

Register transfer Instruction

Register transfer language

- MOV: r1 ← r2 or T: r1 ← r2 means register to register transfer from r2 to r1 on the control signal activation for instruction MOV execution or control signal T
- LD: r1 ← Maddr means load r1 by data bits transfer from memory Maddr on the control signal activation for instruction LD
- ST: Maddr ← r1 means store data bits at memory by transfer from r1 on the control signal activation for instruction ST

Destination first

- Certain processors, for example, INTEL-processor instructions, are represented with destination operand first after the opcode and source operand next
- MOV r1, r2 is when the bits from a register r2 are transferred (copied) into r1
- It means r1 ← r2. The arrow shows the direction (right to left) of transfer from r2 to r1

Destination last

- Other processors, for example, Motorola-processor instructions represent with source operand first after the opcode and destination operand next
- MOV r2, r1 is when bits from a register r2 are transferred (copied) into r1
- It means r2 $\rightarrow$ r1 arrow shows the direction of transfer from r2 to r1

Assumed convention

- In future discussions let us use the convention that unless otherwise stated the destination operand is written first after the opcode and then next source operand

Summary

We Learnt

- A register shown by few characters
- Register transfer shown by left directed or right directed arrow as per convention used
- Control signal or Instruction prefixed followed by colon
- Number of bits in parentheses with the register

End of Lesson 03 on

Register transfers