Chapter 7

Section 7.1 – Control Memory

- The control unit in a digital computer initiates sequences of microoperations
- The complexity of the digital system is derived from the number of sequences that are performed
- When the control signals are generated by hardware, it is *hardwired*
- In a bus-oriented system, the control signals that specify microoperations are groups of bits that select the paths in multiplexers, decoders, and ALUs.

- The control unit initiates a series of sequential steps of microoperations
- The control variables can be represented by a string of 1’s and 0’s called a *control word*
- A *microprogrammed control unit* is a control unit whose binary control variables are stored in memory
- A sequence of microinstructions constitutes a *microprogram*
- The control memory can be a read-only memory
- *Dynamic* microprogramming permits a microprogram to be loaded and uses a writable control memory

- A computer with a microprogrammed control unit will have two separate memories: a main memory and a control memory
- The microprogram consists of microinstructions that specify various internal control signals for execution of register microoperations
- These microinstructions generate the microoperations to:
  - fetch the instruction from main memory
  - evaluate the effective address
  - execute the operation
  - return control to the fetch phase for the next instruction

- The control memory address register specifies the address of the microinstruction
- The control data register holds the microinstruction read from memory
- The microinstruction contains a control word that specifies one or more microoperations for the data processor

- The location for the next microinstruction may, or may not be the next in sequence
- Some bits of the present microinstruction control the generation of the address of the next microinstruction
- The next address may also be a function of external input conditions
- While the microoperations are being executed, the next address is computed in the next address generator circuit (sequencer) and then transferred into the CAR to read the next microinstructions
- Typical functions of a sequencer are:
- Incrementing the CAR by one
- Loading into the CAR and address from control memory
- Transferring an external address
- Loading an initial address to start the control operations

- A clock is applied to the CAR and the control word and next-address information are taken directly from the control memory
- The address value is the input for the ROM and the control work is the output
- No read signal is required for the ROM as in a RAM

- The main advantage of the microprogrammed control is that once the hardware configuration is established, there should be no need for h/w or wiring changes
- To establish a different control sequence, specify a different set of microinstructions for control memory

**Section 7.2 – Address Sequencing**

- Microinstructions are stored in control memory in groups, with each group specifying a **routine**
- Each computer instruction has its own microprogram routine to generate the microoperations
- The hardware that controls the address sequencing of the control memory must be capable of sequencing the microinstructions within a routine and be able to branch from one routine to another

- Steps the control must undergo during the execution of a single computer instruction:
  - Load an initial address into the CAR when power is turned on in the computer. This address is usually the address of the first microinstruction that activates the instruction fetch routine – IR holds instruction
  - The control memory then goes through the routine to determine the effective address of the operand – AR holds operand address
  - The next step is to generate the microoperations that execute the instruction by considering the opcode and applying a **mapping**
  - After execution, control must return to the fetch routine by executing an unconditional branch

- The microinstruction in control memory contains a set of bits to initiate microoperations in computer registers and other bits to specify the method by which the next address is obtained

- Conditional branching is obtained by using part of the microinstruction to select a specific status bit in order to determine its condition
• The status conditions are special bits in the system that provide parameter information such as the carry-out of an adder, the sign bit of a number, the mode bits of an instruction, and i/o status conditions
• The status bits, together with the field in the microinstruction that specifies a branch address, control the branch logic
• The branch logic tests the condition, if met then branches, otherwise, increments the CAR
• If there are 8 status bit conditions, then 3 bits in the microinstruction are used to specify the condition and provide the selection variables for the multiplexer
• For unconditional branching, fix the value of one status bit to be one load the branch address from control memory into the CAR

• A special type of branch exists when a microinstruction specifies a branch to the first word in control memory where a microprogram routine is located
• The status bits for this type of branch are the bits in the opcode
• Assume an opcode of four bits and a control memory of 128 locations
• The mapping process converts the 4-bit opcode to a 7-bit address for control memory
• This provides for each computer instruction a microprogram routine with a capacity of four microinstructions

Subroutines are programs that are used by other routines to accomplish a particular task and can be called from any point within the main body of the microprogram
• Frequently many microprograms contain identical section of code
• Microinstructions can be saved by employing subroutines that use common sections of microcode
• Microprograms that use subroutines must have a provisions for storing the return address during a subroutine call and restoring the address during a subroutine return
• A subroutine register is used as the source and destination for the addresses

Section 7.3 – Microprogram Example

• The process of code generation for the control memory is called microprogramming

Two memory units:
  o Main memory – stores instructions and data
  o Control memory – stores microprogram

Four processor registers
  o Program counter – PC
  o Address register – AR
  o Data register – DR
  o Accumulator register - AC
• Two control unit registers
  o Control address register – CAR
  o Subroutine register – SBR

• Transfer of information among registers in the processor is through MUXs rather than a bus

• Three fields for an instruction:
  o 1-bit field for indirect addressing
  o 4-bit opcode
  o 11-bit address field

• The example will only consider the following 4 of the possible 16 memory instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Opcode</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>0000</td>
<td>AC ← AC + M[EA]</td>
</tr>
<tr>
<td>BRANCH</td>
<td>0001</td>
<td>If (AC &lt; 0) then (PC ← EA)</td>
</tr>
<tr>
<td>STORE</td>
<td>0010</td>
<td>M[EA] ← AC</td>
</tr>
<tr>
<td>EXCHANGE</td>
<td>0011</td>
<td>AC ← M[EA], M[EA] ← AC</td>
</tr>
</tbody>
</table>

• The microinstruction format is composed of 20 bits with four parts to it
  o Three fields F1, F2, and F3 specify microoperations for the computer [3 bits each]
  o The CD field selects status bit conditions [2 bits]
  o The BR field specifies the type of branch to be used [2 bits]
  o The AD field contains a branch address [7 bits]

• Each of the three microoperation fields can specify one of seven possibilities
• No more than three microoperations can be chosen for a microinstruction
• If fewer than three are needed, the code 000 = NOP

```
DR ← M[AR], PC ← PC +1  F2 = 100 and F3 = 101
F1 F2 F3 = 000 100 101
```

• Five letters to specify a transfer-type microoperation
  o First two designate the source register
  o Third is a ‘T’
  o Last two designate the destination register

```
AC ← DR  F1 = 100  = DRTAC
```

• The condition field is two bits to specify four status bit conditions

<table>
<thead>
<tr>
<th>Code</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Always = 1</td>
</tr>
<tr>
<td>01</td>
<td>DR(15)</td>
</tr>
<tr>
<td>10</td>
<td>AC(15)</td>
</tr>
<tr>
<td>11</td>
<td>AC = 0</td>
</tr>
</tbody>
</table>
The branch field is two bits and is used with the address field to choose the address of the next microinstruction

<table>
<thead>
<tr>
<th>Branch</th>
<th>Microoperations</th>
<th>Condition</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>JMP</td>
<td>CAR ← AD if condition =1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CAR ← CAR + 1 else</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>CALL</td>
<td>CAR ← AD, SBR ← CAR + 1 if cond. =1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CAR ← CAR +1 else</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>RET</td>
<td>CAR ← SBR</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>MAP</td>
<td>CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0</td>
<td></td>
</tr>
</tbody>
</table>

Each line of an assembly language microprogram defines a symbolic microinstruction and is divided into five parts

1. The label field may be empty or it may specify a symbolic address. Terminate with a colon
2. The microoperations field consists of 1-3 symbols, separated by commas. Only one symbol from each F field. If NOP, then translated to 9 zeros
3. The condition field specifies one of the four conditions
4. The branch field has one of the four branch symbols
5. The address field has three formats
   a. A symbolic address – must also be a label
   b. The symbol NEXT to designate the next address in sequence
   c. Empty if the branch field is RET or MAP and is converted to 7 zeros

The symbol ORG defines the first address of a microprogram routine

ORG 64 – places first microinstruction at control memory 1000000

The control memory has 128 locations, each one is 20 bits
The first 64 locations are occupied by the routines for the 16 instructions, addresses 0-63

Can start the fetch routine at address 64
The fetch routine requires the following three microinstructions (locations 64-66)

AR ← PC
DR ← M[AR], PC ← PC +1
AR ← DR(0-10), CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0

ORG 64
Fetch: PCTAR  U  JMP  NEXT
READ, INCPC  U  JMP  NEXT
DRTAR  U  MAP

<table>
<thead>
<tr>
<th>Address</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>CD</th>
<th>BR</th>
<th>AD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000000</td>
<td>110</td>
<td>000</td>
<td>000</td>
<td>00</td>
<td>00</td>
<td>1000001</td>
</tr>
<tr>
<td>1000001</td>
<td>000</td>
<td>100</td>
<td>101</td>
<td>00</td>
<td>00</td>
<td>1000010</td>
</tr>
<tr>
<td>1000010</td>
<td>101</td>
<td>000</td>
<td>000</td>
<td>00</td>
<td>11</td>
<td>0000000</td>
</tr>
</tbody>
</table>