Advanced Computer Architecture

Lecture No. 22

Reading Material

Vincent P. Heuring&Harry F. Jordan	Chapter 5
Computer Systems Design and Architecture	5.3

Summary

- Microprogramming
- Working of a General Microcoded Controller
- Microprogram Memory
- Generating Microcode for Some Sample Instructions
- Horizontal and Vertical Microcode Schemes
- Microcoded 1-bus SRC Design
- The SRC Microcontroller

Microprogramming

In the previous lectures, we have discussed how to implement logic circuitry for a control unit based on logic gates. Such an implementation is called a hardwired control unit. In a micro programmed control unit, control signals which need to be generated at a certain time are stored together in a control word. This control word is called a microinstruction. A collection of microinstructions is called a microprogram. These microprograms generate the sequence of necessary control signals required to process an instruction. These microprograms are stored in a memory called the control store.
As described above microprogramming or microcoding is an alternative way to design the control unit. The microcoded control unit is itself a small stored program computer consisting of

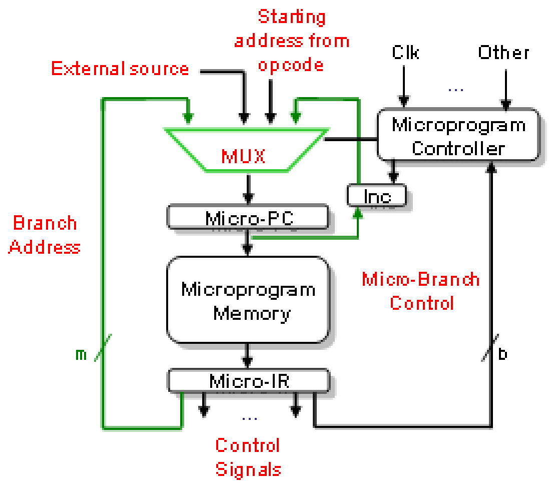
- Micro-PC
- Microprogram memory
- Microinstruction word

Comparison of hardwired and microcoded control unit

Hardwired Control Unit	**Microcoded Control Unit**
The relationship between control inputs and control outputs is a series of Boolean functions.	The control signals here are stored as words in a microcode memory.
Hardwired control units are generally faster.	Microcode units simplify the computer logic but it is comparatively slower.

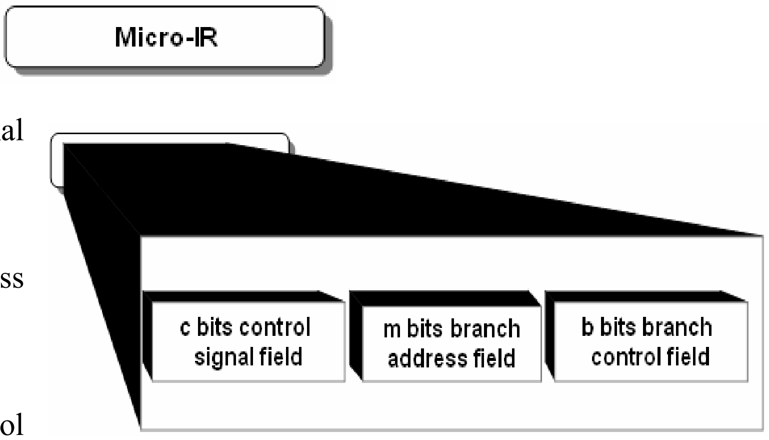
Working of a general microcoded controller

A microcoded controller works in the same way as a small general purpose computer.
1. Fetch a micro-instruction and increment micro-PC.
2. Execute the instruction present in micro-IR.
3. Fetch the next instruction and so on…



The microcoded control unit is like a small computer in itself. It consists of a microprogram memory, which is read using a micro program counter. The micro PC is controlled by the microprogram controller. Values of the micro PC depends on a 4 to 1 MUX. The source may be the incremented micro PC value, or a calculated branch value, or a value derived by decoding an opcode for an instruction. The microprogram memory writes the control word at the chosen address into the micro instruction register. This control word is basically the set of all the control signals needed to execute the instruction at that particular instant.

Fields in the micro instruction


Micro-IR
c bits control signal field
m bits branch address field
b bits branch control field

C Bits
These form the control signal field

M Bits
These form the branch address field

B Bits
These form the branch control field.

Loading the micro-PC
The micro-PC can be loaded from one of the four possible sources
- **Simple increment** Steps sequentially from microinstruction to microinstruction
- **Lookup table** A lookup table maps the opcode field to the starting address of the microcode routine that generates control signals.

- **External source** Initializes micro-PC to begin an operation e.g. interrupts service, reset etc.
- **Branch addresses** Jumps anywhere in the microprogram memory on the basis of conditional or unconditional branch.

Microprogram Memory

- This small memory contains microroutines for all the instructions in the ISA
- The micro-PC supplies the address and it returns the control word stored at that address
- It is much faster and smaller than a typical main memory

Layout of a typical microprogram memory

Micro-Address	Memory Contents
0	Microcode for instruction fetch
...	Microcode for load instruction
...	Microcode for add instruction
...	Microcode for br instruction
2^n-1	Microcode for reset instruction

Generating Microcode for Some Sample Instructions

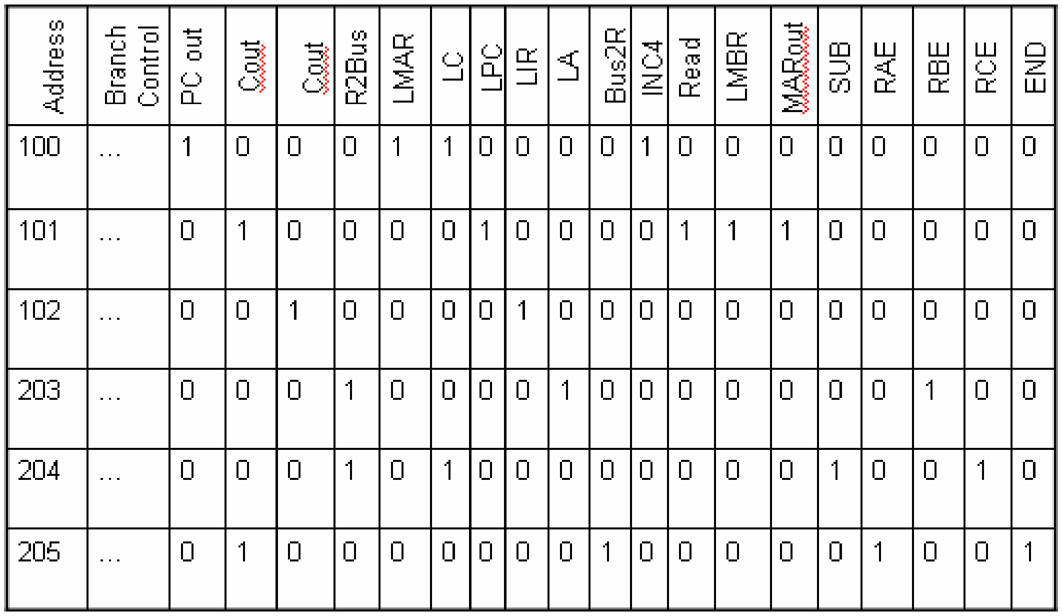
- The control word for an instruction is used to generate the equivalent microcode sequence
- Each step in RTL corresponds to a microinstruction executed to generate the control signals.

Each bit in the control words in the microprogram memory represents a control signal. The value of that bit decides whether the signal is to be activated or not.

Example: Control Signals for the sub Instruction

The first three addresses from 100 to 102 represent microcode for instruction fetch and the last three addresses from 203 to 205 represent microcode for sub instruction. In the first cycle at address 100, the control signal PCout, LMAR, LC, and INC4 are activated and all other signals are deactivated. All these control signals are for the SRC processor.

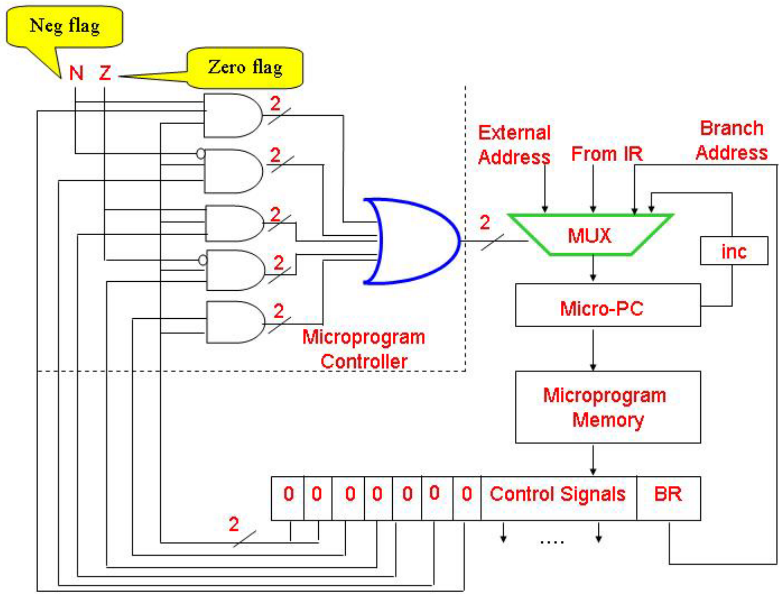
So, if the micro-PC contains 100, the contents of microprogram memory are copied into the micro IR. This corresponds to the structural RTL description of the T0 clock during instruction fetch phase. In the same way, the content of address 101 corresponds to T1, and the content of address 102 corresponds to T2.



Address	Branch Control	PC out	Cout	Cout	R2Bus	LMAR	LC	LPC	LIR	LA	Bus2R	INC4	Read	LMBR	MARout	SUB	RAE	RBE	RCE	END
100	…	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0
101	…	0	1	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0
102	…	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
203	…	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
204	…	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0
205	…	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1

Microprogram Controller functions: Branching and looping

- Microprogram controller controls the sequence of the flow of microinstructions.
- The inputs to the microcontroller are from the branch control fields specified in the microcode word.
- Its output controls the 4 to 1 multiplexer inside the microcoded control unit.
- It implements conditional execution and both conditional and unconditional branch



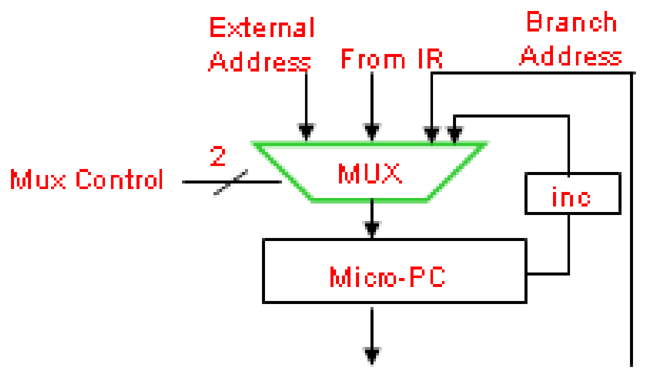
If a branch instruction is encountered within the microprogram hardwired logic selects the branch address as the source of micro-PC using 4 to 1 mux. This hardwired logic caters for all branch instructions including branch if zero.

4-1 Multiplexer

The multiplexer supplies one of the four possible values to the micro-PC
The incremented value of the micro-PC is used when dealing with the normal flow of microinstructions.
The opcode from the instruction is used to set the micro-PC when a microroutine is initially being loaded.

External address is used when it is required to reset the microprogram controller.
Branch address is set into the micro-PC when a branch microinstruction is encountered.

Mux Control	Select
00	Increment micro-PC
01	Opcode from IR
10	External address
11	Branch address



How to form a branch

- A branch can be implemented by choosing one alternative from each of the following two lists.
- This scheme provides flexibility in choosing branches as we can form any combination of conditions and addresses.

Condition
unconditional
not zero
zero
positive
negative

Address
From IR
External Address
Branch Address

Microcode Branching Examples

Following is an example of branch instructions in microcode

Address	Mux COntrol	Branch	brnz	brz	brp	brn	Control Signals	Branch Address	Branching Action	Equivalent C construct
400	00	0	0	0	0	0	…	xxx	No branch,goto next address in sequence-401	{…};
401	01	1	0	0	0	0	…	xxx	To the address supplied by opcode	{…}; goto initial address;
402	10	0	0	1	0	0	…	xxx	To external address if Z flag is set	{…}; if Z then goto Ext. Add.
403	11	0	0	0	0	1	…	200	To 200 if N flag is set, else to 404	{…}; if N then goto Label1;
404	11	0	0	0	1	0	000	406	To 406 if N is false, else to 405	While (N) {...};
405	11	1	0	0	0	0	…	404	Branch to 404	While contd…

Similarity between microcode and high level programs

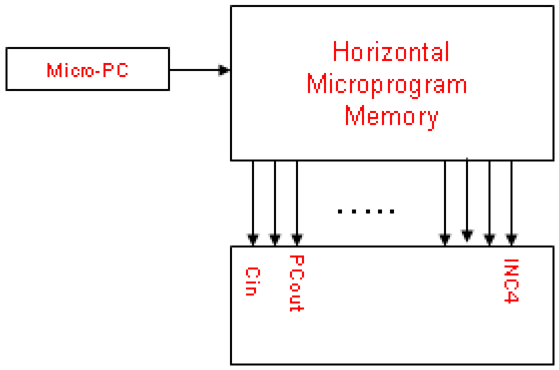
- Any high level construct such as if-else, while, repeat etc. can be implemented using microcode
- A variety of microcode compilers similar to the high level compilers are available that allow easier programming in microcode
- This similarity between high level language and microcode simplifies the task of controller design.

Horizontal and vertical microcode schemes

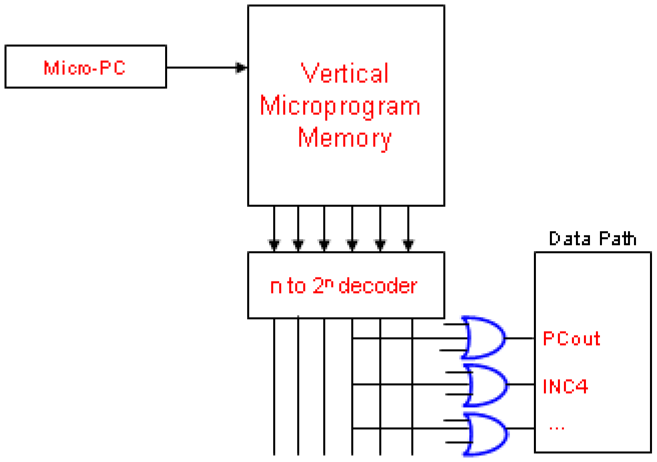
In horizontal microcode schemes, there are no intermediate decoders and the control word bits are directly connected to their destination i.e. each bit in the control word is directly connected to some control signal and the total number of bits in the control word is equal to the total number of control signals in the CPU.
Vertical microcode schemes employ an extra level of decoding to reduce the control word width. From an n bit control word we may have 2^n bit signal values.

However, a completely vertical scheme is not feasible because of the high degree of fan out.

Horizontal Microcode Scheme



Vertical Microcode Scheme

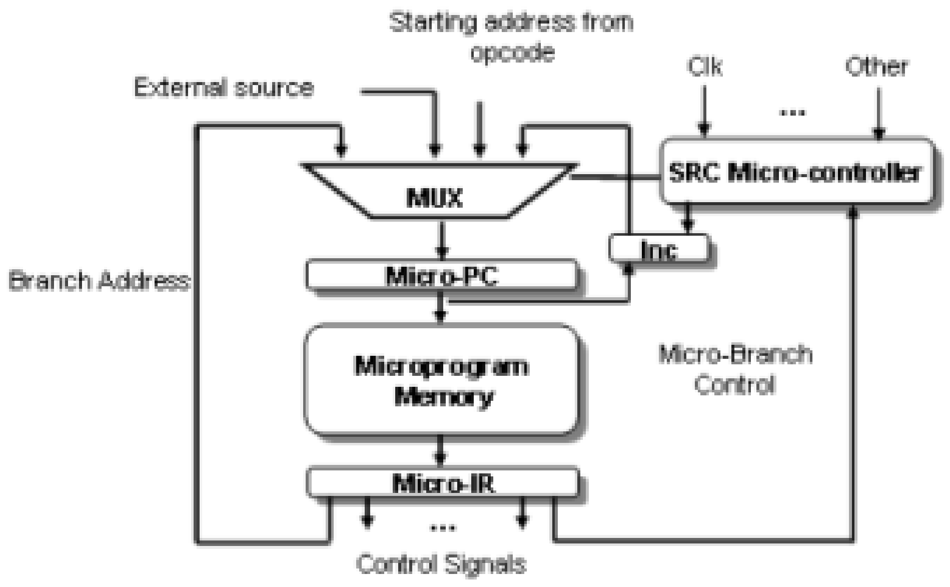


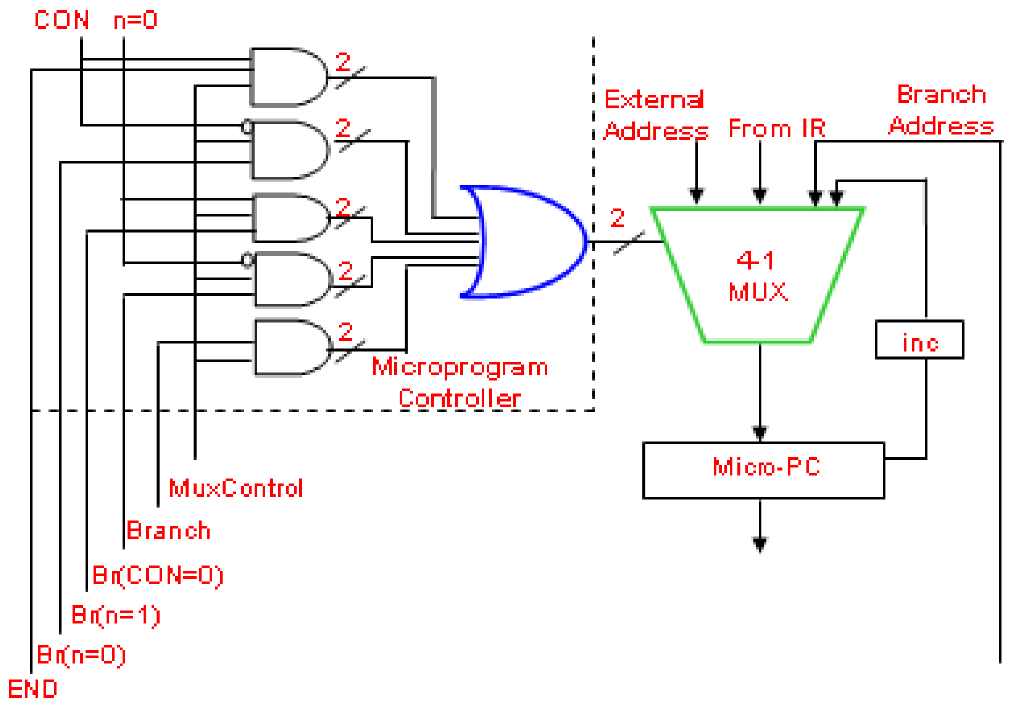
Microcoded 1-bus SRC design

In the SRC the bits from the opcode in the instruction register are decoded to fetch the address of the suitable microroutine from the microprogram memory. The microprogram controller for the SRC microcoded control unit employs the logic for handling exceptions and reset process. Since the SRC does not have any condition codes, we use the CON and n signals instead of N and Z flags to control branches in case of branch if equal to zero or branch if less than instructions.

The SRC Microprogram Controller

• The microprogram controller for the SRC microcoded control unit employs the logic for handling exceptions and reset process
• Since the SRC does not have any condition codes, we use the CON and n signals instead of N and Z flags to control branches





Microcode for some SRC instructions

Address	MuxControl	Branch	Br(CON=0)	Br(n=1)	Br(n=0)	End	PCout	LMAR	Control Signals	Branch Address	RTL
300	00	0	0	0	0	0	1	1	…	xxx	MAR← PC: C ← PC + 4;
301	00	0	0	0	0	0	0	0	…	xxx	MBR← M[MAR]: PC ← C;
302	01	1	0	0	0	0	0	0	…	xxx	IR,Micro-PC←MBR<31…27>;
400	00	0	0	0	0	0	0	0	…	xxx	A ← R[rb];
401	00	0	0	0	0	0	0	0	…	xxx	C ← A + R[rc];
402	11	1	0	0	0	1	0	0	…	300	R[ra] ← C; Micro-PC ← 300;

Assume the first control word at address 300. The RTL of this instruction is MAR ← PC combined with C ←PC+4. To facilitate these actions the PCout signal bit and the LMAR signal bit are set to one, so that the value of the PC may be written to the internal processor bus and written onto the MAR. The instructions at 300, 301 and 302 form the microcode for instructions fetch. If we examine the RTL we can see all the functionality of the fetch instruction. The value of PC is incremented, the old value of PC is sent to memory, the instruction from the sent address is loaded into memory buffer register. Then the opcode of the fetched instruction is used to invoke the appropriate microroutine.

Alternative approaches to microcoding

- Bit ORing
- Nanocoding
- Writable Microprogram Memory
- Subroutines in Microprogramming