

## ECE/CS 3724 - Microprocessors

- Everything relating to the class is posted on:  
<http://www.ece.msstate.edu/~reese/EE3724>
- Most of the course lecture notes are linked to WWW page -- print them out before class and bring them to lecture
- Be sure to read the class Policy/Syllabus, and follow the link to the EE 3724 LAB page
- Lots of material in this class - don't fall behind!

BR 6/00

1

---

---

---

---

---

---

---

## Approaches to Digital System Design

- In Digital Devices, you learned how to create a logic network (Flip-flops + combinational gates) to solve a problem
  - The logic network was SPECIFIC to the problem. To solve a different problem, needed a different logic network
- Another approach is to design a logic network that can used to solve many different problems
  - This *general purpose* logic network might not be as efficient (speed, cost) as a special purpose logic network, but hopefully can be used to solve multiple problems!

BR 6/00

2

---

---

---

---

---

---

---

## A Computer!!

- A **Computer** is a digital system whose operation can be specified via a **Program**.
  - Changing the program changes the computer behavior! (solves a different problem!!!).
- A Program is simply a sequence of binary codes that represent instructions for the computer. The Program is stored in a **Memory**.
- External inputs to the Computer can also alter the behavior the computer. The computer will have Outputs that can be set/reset via program instructions.
  - These external inputs/output are know as the I/O section of the computer.

BR 6/00

3

---

---

---

---

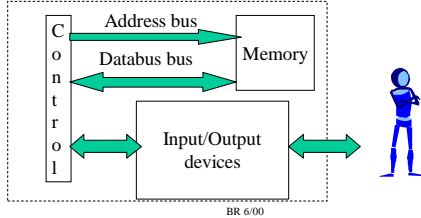
---

---

---

## Components of any Computer System

- Control – logic that controls fetching/execution of instructions
- Memory – area where instructions/data are stored
- Input/Output – external interaction with computer



4

## Problem Definition

Build a Digital System based upon your Social Security number (SSN).

The Digital System will have one external input called ODD.

If ODD is true, then the system will reset to display the LEFTMOST odd digit in your SSN, and then the output will sequence over the odd digits in your SSN, skipping over the even digits.

If ODD is false, then the system will reset to display the LEFTMOST even digit in your SSN, and then the output will sequence over the even digits in your SSN, skipping over the odd digits.

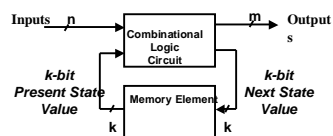
BR 6:00

5

## Two Approaches for Solving this Problem

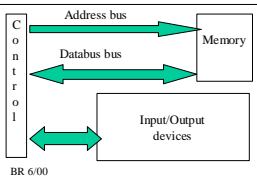
Finite State Machine

Will only work for one SSN sequence

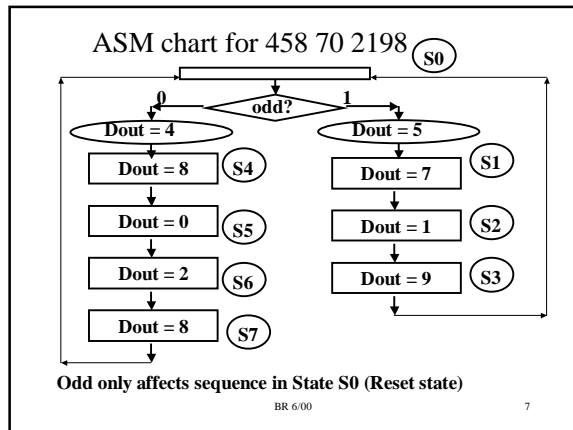


Computer System

Will only work for any SSN sequence, change program to change sequence



6




---

---

---

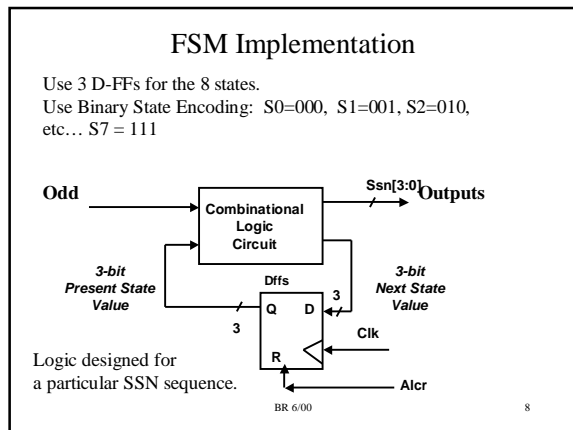
---

---

---

---

---




---

---

---

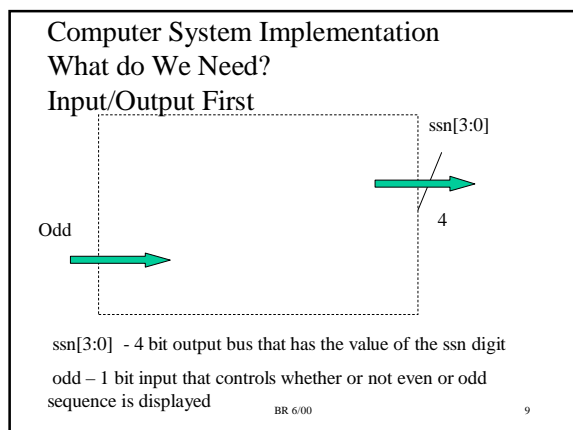
---

---

---

---

---




---

---

---

---

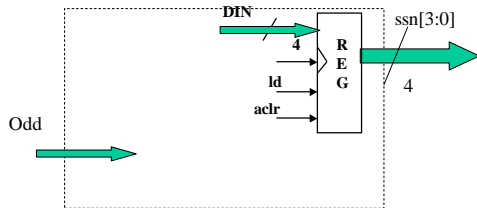
---

---

---

---

### Register for holding SSN output value



Register loads DIN on rising clock edge when ld = 1.

Aclr is an asynchronous clear.

BR 6/00

10

---

---

---

---

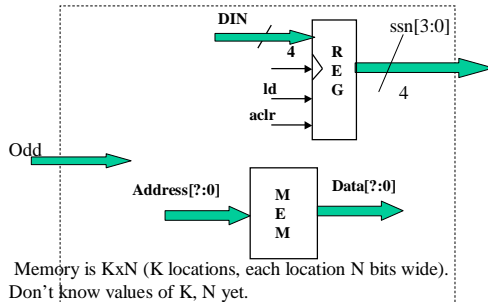
---

---

---

---

### Memory for holding instructions



Memory is KxN (K locations, each location N bits wide).  
Don't know values of K, N yet.

BR 6/00

11

---

---

---

---

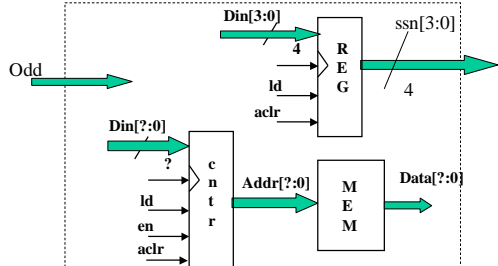
---

---

---

---

### Register for specifying address – Use counter



Counter increments on rising clock edge when en = 1.

Loads on rising clock edge when ld = 1. Aclr is an asynchronous clear.

BR 6/00

12

---

---

---

---

---

---

---

---

## What Instructions do we need?

Start:

If (Odd = 1) goto Odd\_start

Even\_start:

output even digit #1

output even digit #2

.....

output last even digit

goto Start

Odd\_start:

output odd digit #1

output odd digit #2

.....

output last odd digit

goto Start

← Psuedo Code for operations

BR 6/00

13

---

---

---

---

---

---

---

---

## Needed Instructions

1. *Jc location* Jump conditionally  
If odd = 1, then jump to location (counter set equal to specified location).  
If odd = 0, then fetch next instruction (counter increments by 1).
2. *Jmp location* Jump unconditional  
Fetch next instruction from location (counter loaded with specified location).
3. *out data*  
load output register with data. Used for setting the ssn[3:0] value.

BR 6/00

14

---

---

---

---

---

---

---

---

## Instruction Encoding

The binary encoding for instructions is usually divided into different fields; with each field representing part of the information needed by the instruction.

Our instructions require two fields: *Operation Code* and *Data*

Opcode		Data
--------	--	------

How many bits for the Opcode? Have 3 instructions, need at least 2 bits! (2 bits can encode  $2^2$  items)

How many bits for Data? The data field must specify the 4 bits for the SSN number, and also specify a memory location. For now, lets use 4 bits for data. Instruction is 6 bits total.

I5 I4 I3 I2 I1 I0

Opcode		Data
--------	--	------

BR 6/00

15

---

---

---

---

---

---

---

---

### Instruction Table

	I5 I4	I3 I2 I1 I0
JMP location	0 0	4-bit location
JC location	0 1	4-bit location
OUT data	1 0	4-bit data

Note that Opcode = 11 is unused.

Also, the opcode assignments were arbitrary; we could have easily chosen some other assignment

(such as OUT=00, JC=00, JMP=01)

BR 6/00

16

### A Program for SID = 458 70 2198

```

Start:   JC   odd_start   ; jmp only if odd input=1
         OUT  4
         OUT  8
         OUT  0
         OUT  2
         OUT  8
         JMP  Start
Odd_start: OUT  5
         OUT  7
         OUT  1
         OUT  9
         JMP  Start
    
```

BR 6/00

17

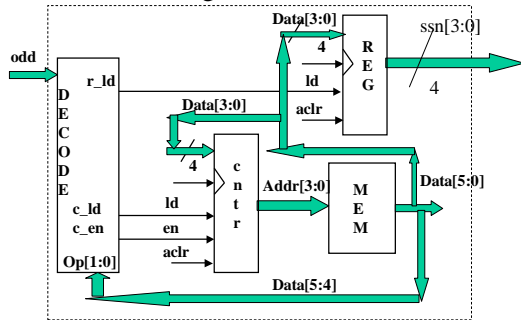
### Convert Program to Binary, Put in Memory

Mem Location	Mem Contents	Instruction
00	01 0111	Start: JC Odd_start (loc 7)
01	10 0100	OUT 4
02	10 1000	OUT 8
03	10 0000	OUT 0
04	10 0010	OUT 2
05	10 1000	OUT 8
06	00 0000	JMP Start (loc 0)
07	10 0101	Odd_start: OUT 5
08	10 0111	OUT 7
09	10 0001	OUT 1
0A	10 1001	OUT 9
0B	00 0000	JMP Start (loc 0)

BR 6/00

18

## Add Decode Logic to Execute Instructions



BR 6/00

19

## What is Decode Logic?

Decode logic controls *count* register, *out* register based on Op code value ( $op[1:0] = Data[5:4]$ ).

When does *out* register get loaded? When  $OP = 10!!$  (OUT instruction):

VHDL:

```
r_ld <= '1' when (op = "10") else '0';
```

When does Counter Load? When JMP instruction ( $OP=00$ ) or when JC instruction and Odd = '1'!!!!

```
c_ld <= '1' when (op = "00" or (op = "01" and odd = '1')) else '0';
```

When does counter increment? When NOT Loading!!

```
c_en <= not (c_ld);
```

BR 6/00

20

## Decode Boolean Equations

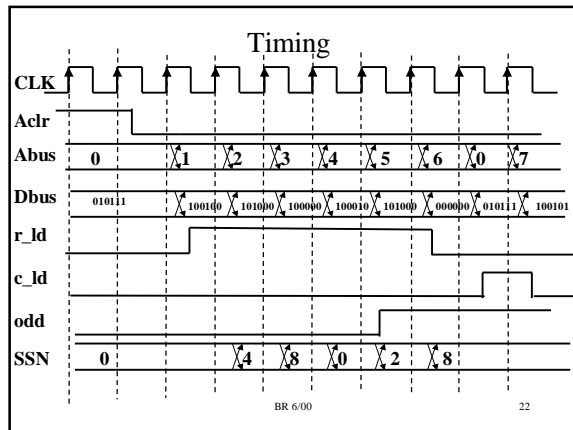
```
r_ld <= op(1) -- don't really need op(0)
```

```
c_ld <= ((not op(1)) and (not op(0))) or  
         ((not op(1)) and op(0) and odd);
```

```
c_en <= not (c_ld);
```

BR 6/00

21




---

---

---

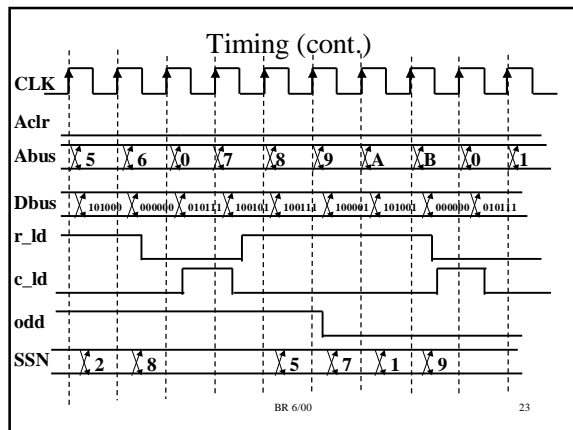
---

---

---

---

---




---

---

---

---

---

---

---

---

### Comments

- Notice that the *aclr* line forces the processor to fetch its **first** instruction from location 0.
  - All processors have a RESET line like this to force the first instruction fetch from a particular location.
- Notice that execution never stops!!! Processor is always fetching, executing instructions!
- Called the *Fetch,Execute* loop.
- Must make sure that memory is loaded with valid instructions BEFORE execution starts!!!

---

---

---

---

---

---

---

---



### Instruction Pointer

- The counter in this processor is a special purpose register that exists in one form or another in *every* processor
- Usually is called the *Instruction Pointer* (IP) register or *Program Counter* (PC) register.
- This register contains the address of the next instruction to be fetched.
  - Normal operation is to fetch very next instruction in memory
  - Jump instructions change the IP value so that fetch occurs from some non-sequential memory location

BR 6/00

25

---

---

---

---

---

---

---

### Implementation Comparisons

- FSM Implementation
  - Only 3 D-FFs + combinational logic
  - Will only do one SSN sequence
  - Will operate at a faster clock rate than Processor implementation because of simpler logic
- Processor Implementation
  - Many more gates needed than FSM implementation
  - Will execute at a slower clock rate than FSM
  - General purpose: can implement any SSN sequence by simply changing program.
- MANY applications are better suited for implementation by general purpose digital systems (Processors) than by dedicated logic

BR 6/00

26

---

---

---

---

---

---

---

### Vocabulary

- *Address bus* – input bus to memory device specifying location of data to read/write
- *Data bus* – input/output bus to memory device containing data value being read or written.
- *Instruction Pointer* – special register in a processor specifying address of next instruction to be executed.
- *Instruction Mnemonic* – the ascii representation of an instruction (I.e., OUT 4).
- *Machine Code* – the binary representation of an instruction (I.e OUT 4 = 010100)

BR 6/00

27

---

---

---

---

---

---

---

### Vocabulary (cont.)

- *Operation code (Op code)* – the part of the machine code for an instruction that tells what the instruction is ( JMP = 00).
- *Assembly* – the process of converting instructions to their machine code representation  
OUT 4 → 10 0100
- *Disassembly* – the process of converting machine code to its instruction mnemonic  
10 0100 → OUT 4
- *Fetch/Execute* - what processors do all day long (fetch instruction from memory, execute it).

BR 6/00

28

---

---

---

---

---

---

---

### How are Commercial Processors different from SSN Processor?

- SSN processor had 4-bit registers. Com. processors have registers with widths from 8 bits to 128 bits wide.
- SSN processor has 2 registers. Com. proc have many registers, some general purpose, some special purpose.
- SSN processor has 3 instructions. Com. Proc have 10's to a few hundred instructions (arithmetic, logical, control, Input/output, data movement, etc).
- SSN processor could address 16 memory locations. Com. Proc can address billions of memory locations.
- SSN processor can be implemented in a few 10's of gates. Com. Processors can take millions of gates to implement.

BR 6/00

29

---

---

---

---

---

---

---

### What do you need to know?

- Differences between specific logic networks and general purpose logic networks for digital systems.
- Basics of a computer system
- Logic Structure, timing of our SSN sequence processor
- Instruction assembly, disassembly, execution of SSN sequence processor
- Vocabulary

BR 6/00

30

---

---

---

---

---

---

---