CS3410 Prelim 1 Review

2014 Spring

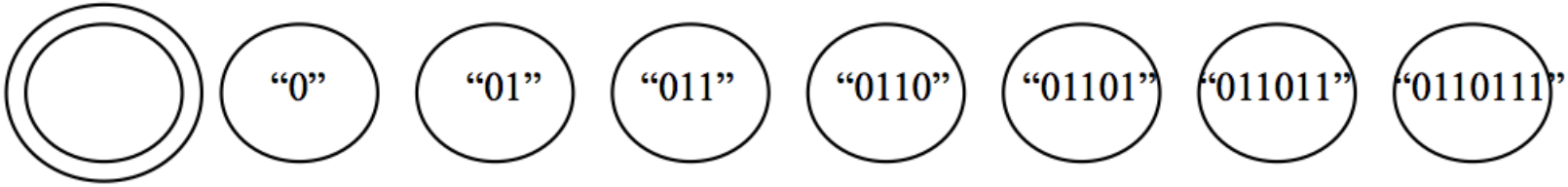
FSM

Design a circuit (with a finite state machine) for finding the *each* occurrence of a specific pattern, 0110111 or 0110 in a very long stream of binary numbers.

Specifically, your circuit will process a very long stream of binary numbers, b0b1b2b3.... You do not know how long the list will be. Your circuit will be given the i-th binary number during the i- th clock cycle via the input B. If you have not seen the pattern yet, then Found should be 0; otherwise, if bi-6bi-5bi-4bi-3bi-2bi-1bi = 0110111 or bi-3bi-2bi-1bi = 0110 then your circuit should set the output Found to 1. After finding the pattern 0110111, your circuit should go back to either the init state or the first state after init so that the pattern can be recognized again.

FSM

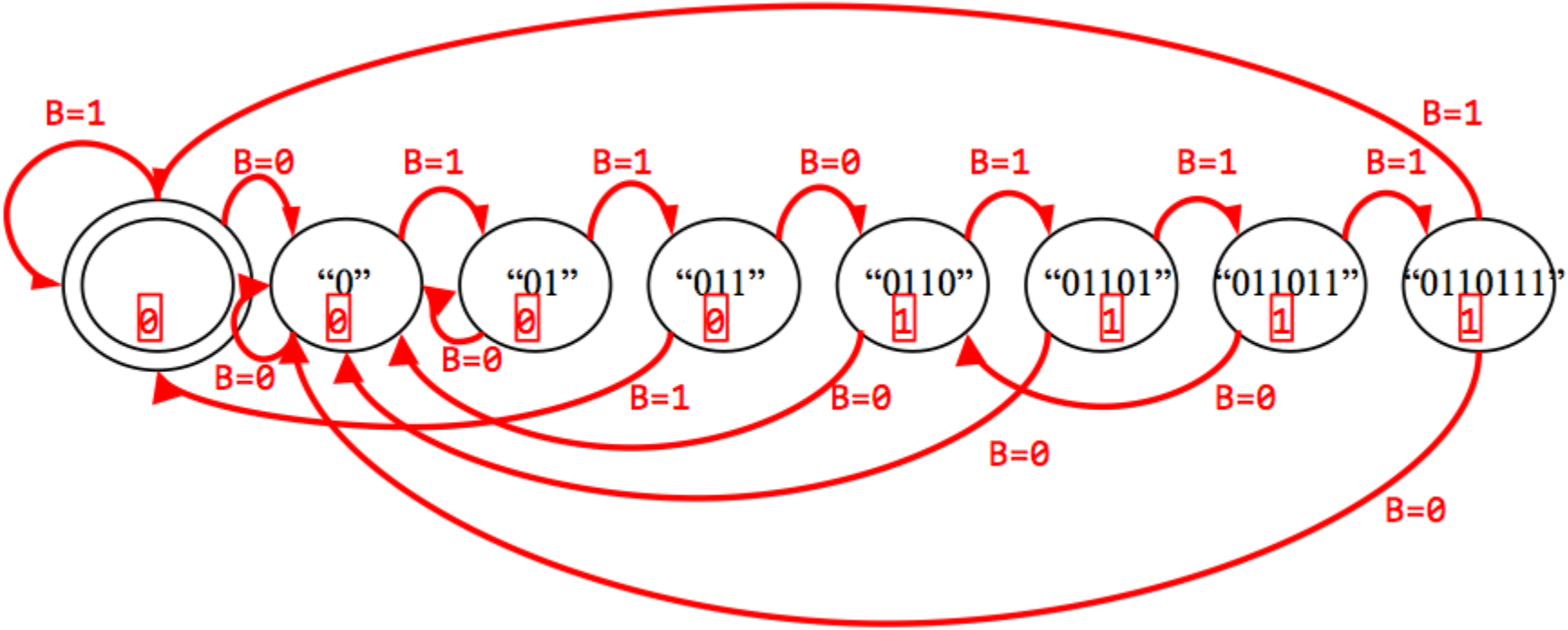
Design a circuit (with a finite state machine) for finding the *each* occurrence of a specific pattern, 0110111 or 0110 in a very long stream of binary numbers.



Complete the finite state machine diagram above by drawing transition arrows, and clearly labeling all transitions, inputs, and outputs. You are to build a *Moore-style* finite state machine with a single bit B as input and a single bit Found as output. You should label each state transition with the expected value of input B and each state with the value of Found.

Hints: 1) Your circuit should work for list of numbers like 001101111 and 011011011101. 2) Recall that for a Moore machine, the outputs depends only on the current state, and hence the output is written *in the state circles*.

Design a circuit (with a finite state machine) for finding the *each* occurrence of a specific pattern, 0110111 or 0110 in a very long stream of binary numbers.



Performance

Name the three principal components of the runtime for a program that we discussed in class. How do they combine to yield runtime?

Performance

Name the three principal components of the runtime for a program that we discussed in class. How do they combine to yield runtime?

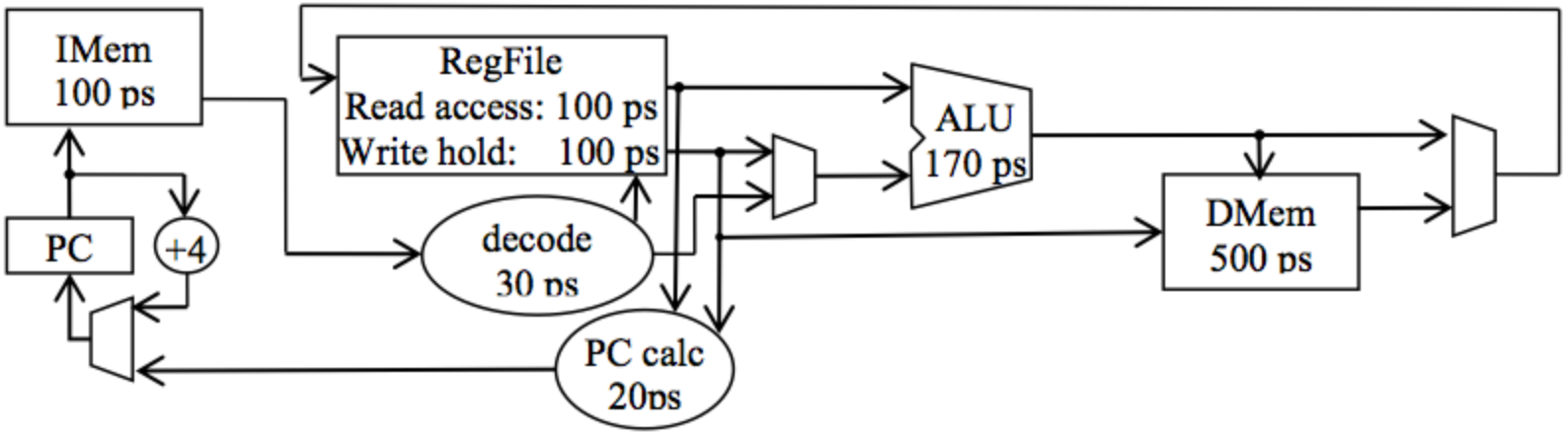
Instruction Count, CPI, and Clock Period (or Rate)

Runtime = Instruction count x CPI x Clock Period

- or -

$$\frac{\text{Instruction count (instr)} \times \text{CPI (cycles/instr)}}{\text{Clock Rate (cycles/sec)}}$$

Performance



running a workload consisting of the following mix of instructions:

Arithmetic / Logic	60%
Memory Load/ Store	25%
Branches	15%

Performance

1. Consider a **non-pipelined single-cycle** version of the datapath. Identify the critical path and determine the optimal clock rate. What is the CPI and MIPS?

2. **non-pipelined multi-cycle**

3. **five stage pipeline**

Performance

1. Consider a **non-pipelined single-cycle** version of the datapath. Identify the critical path and determine the optimal clock rate. What is the CPI and MIPS?

Memory path is the critical path

1000 ps is the optimal (i.e. minimum) clock period (this is the same as a clock frequency of 1 GHz)

CPI = 1 (i.e. one cycle per instruction given a single-cycle processor) MIPS = 1000 MIPS (i.e. 1000 x million [106] instructions per second)

Performance

2. **non-pipelined multi-cycle**

arithmetic takes 500 ps, so 2 cycles
memory takes 1000 ps, so 4 cycles
all branches take 250 ps, so 1 cycles

*CPI = weighted average = 60% * 2 + 25% * 4 + 15% * 1 = 1.2 + 1 + 0.15 = 2.35*

Note 4 GHz = 4000 MHz

$$MIPS = \frac{4000\ MHz}{2.35\ CPI} = \frac{4000 \times 10^6\ cycles\ per\ second}{2.35\ cycles\ per\ instruction}$$
$$= 1702 \times 10^6\ instructions\ per\ second$$
$$= 1702\ MIPS$$

Performance

3. **five stage pipeline**

All instructions take 5 cycles if we ignore delays due to branching!

CPI = 1

$$MIPS = \frac{2000\ MHz}{1\ CPI} = 2000\ MIPS$$

Data Hazards

```
ADDIU  r1, r0, 0x04
LW r3, 0(r1)
LW r4, 4(r1)
SLL r4, r4, 2
ADDU r4, r4, r3
SW r4, 8(r1)
```

Assume that a 5-stage pipelined MIPS processor does NOT account for any pipeline data hazard (and cannot write to and read from the register file during the same cycle). Rewrite the above instructions putting in NOPs so that no errors will occur. Do not reorder instructions. What is the minimal number of cycles it would take for the corrected code to completely clear the MIPS pipeline?

Data Hazards

```
ADDIU  r1, r0, 0x04
NOP
NOP
NOP
LW r3, 0(r1)                    ; because r1
LW r4, 4(r1)
NOP
NOP
NOP
SLL r4, r4, 2                   ; because r4
NOP
NOP
NOP
ADDU r4, r4, r3                 ; because r4
NOP
NOP
NOP
SW r4, 8(r1)                    ; because r4
```

It will take 22 cycles for the code to completely clear the MIPS pipeline

Data Hazards

Assume the pipeline now has a hazard detection unit and can automatically insert the required NOPs (stalls) for correct execution.

Fill in the multi-clock cycle graph with the required stalls for the original instructions to execute correctly. What is the minimal number of cycles it will take for the code to completely clear the MIPS pipeline if the processor automatically inserts the required NOPs

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ADDIU	IF	ID	Ex	M	W																		

Data Hazards

Assume the pipeline now has a hazard detection unit and can automatically insert the required NOPs (stalls) for correct execution.

Fill in the multi-clock cycle graph with the required stalls for the original instructions to execute correctly. What is the minimal number of cycles it will take for the code to completely clear the MIPS pipeline if the processor automatically inserts the required NOPs

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ADDIU	IF	ID	Ex	M	W																		
LW		*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*														
LW			*IF*	*IF*	*IF*	*IF*	*ID*	*Ex*	*M*	*W*													
SLL							*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*									
ADDU								*IF*	*IF*	*IF*	*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*					
SW												*IF*	*IF*	*IF*	*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*	

It will take 22 cycles for the code to completely clear the MIPS pipeline

Data Hazards

Assume the pipeline now has a hazard detection unit and can automatically insert the required NOPs (stalls) for correct execution.

Fill in the multi-clock cycle graph with the required stalls for the original instructions to execute correctly. What is the minimal number of cycles it will take for the code to completely clear the MIPS pipeline if the processor automatically inserts the required NOPs

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ADDIU	IF	ID	Ex	M	W																		
LW		*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*														
LW			*IF*	*IF*	*IF*	*IF*	*ID*	*Ex*	*M*	*W*													
SLL							*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*									
ADDU								*IF*	*IF*	*IF*	*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*					
SW												*IF*	*IF*	*IF*	*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*	

It will take 22 cycles for the code to completely clear the MIPS pipeline

Stalled in ID waiting for WB to complete

Data Hazards

Assume the pipeline now has a hazard detection unit and can automatically insert the required NOPs (stalls) for correct execution.

Fill in the multi-clock cycle graph with the required stalls for the original instructions to execute correctly. What is the minimal number of cycles it will take for the code to completely clear the MIPS pipeline if the processor automatically inserts the required NOPs

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ADDIU	IF	ID	Ex	M	W																		
LW		*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*														
LW			*IF*	*IF*	*IF*	*IF*	*ID*	*Ex*	*M*	*W*													
SLL							*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*									
ADDU								*IF*	*IF*	*IF*	*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*					
SW												*IF*	*IF*	*IF*	*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*	

It will take 22 cycles for the code to completely clear the MIPS pipeline

Stalled in IF waiting for ID to clear

Data Hazards

Assume the pipeline now has a hazard detection unit and can automatically insert the required NOPs (stalls) for correct execution.

Fill in the multi-clock cycle graph with the required stalls for the original instructions to execute correctly. What is the minimal number of cycles it will take for the code to completely clear the MIPS pipeline if the processor automatically inserts the required NOPs

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ADDIU	IF	ID	Ex	M	W																		
LW		*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*														
LW			*IF*	*IF*	*IF*	*IF*	*ID*	*Ex*	*M*	*W*													
SLL							*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*									
ADDU								*IF*	*IF*	*IF*	*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*					
SW												*IF*	*IF*	*IF*	*IF*	*ID*	*ID*	*ID*	*ID*	*Ex*	*M*	*W*	

It will take 22 cycles for the code to completely clear the MIPS pipeline

Waiting for IF to clear

Data Hazards

Assume that we have a pipeline that has both a forwarding unit and hazard detection unit: The forwarding unit can automatically resolve data hazards by forwarding values between stages.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ADDIU	IF	ID	Ex	M	W																		

Data Hazards

Assume that we have a pipeline that has both a forwarding unit and hazard detection unit: The forwarding unit can automatically resolve data hazards by forwarding values between stages.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ADDIU	IF	ID	Ex	M	W																		
LW		*IF*	*ID*	*Ex*	*M*	*W*																	
LW			*IF*	*ID*	*Ex*	*M*	*W*																
SLL				*IF*	*ID*	*ID*	*Ex*	*M*	*W*														
ADDU					*IF*	*IF*	*ID*	*Ex*	*M*	*W*													
SW							*IF*	*ID*	*Ex*	*M*	*W*												

It will take 11 cycles for the instructions to completely clear the MIPS pipeline

Data Hazards

Assume that we have a pipeline that has both a forwarding unit and hazard detection unit: The forwarding unit can automatically resolve data hazards by forwarding values between stages.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ADDIU	IF	ID	Ex	M	W																		
LW		*IF*	*ID*	*Ex*	*M*	*W*																	
LW			*IF*	*ID*	*Ex*	*M*	*W*																
SLL				*IF*	*ID*	*ID*	*Ex*	*M*	*W*														
ADDU					*IF*	*IF*	*ID*	*Ex*	*M*	*W*													
SW							*IF*	*ID*	*Ex*	*M*	*W*												

It will take 11 cycles for the instructions to completely clear the MIPS pipeline

Not ready until MEM can forward

Data Hazards

Assume that we have a pipeline that has both a forwarding unit and hazard detection unit: The forwarding unit can automatically resolve data hazards by forwarding values between stages.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ADDIU	IF	ID	Ex	M	W																		
LW		*IF*	*ID*	*Ex*	*M*	*W*																	
LW			*IF*	*ID*	*Ex*	*M*	*W*																
SLL				*IF*	*ID*	*ID*	*Ex*	*M*	*W*														
ADDU					*IF*	*IF*	*ID*	*Ex*	*M*	*W*													
SW							*IF*	*ID*	*Ex*	*M*	*W*												

It will take 11 cycles for the instructions to completely clear the MIPS pipeline

Waiting for ID to clear

Data Hazards

Assume that we have a pipeline that has both a forwarding unit and hazard detection unit: The forwarding unit can automatically resolve data hazards by forwarding values between stages.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ADDIU	IF	ID	Ex	M	W																		
LW		*IF*	*ID*	*Ex*	*M*	*W*																	
LW			*IF*	*ID*	*Ex*	*M*	*W*																
SLL				*IF*	*ID*	*ID*	*Ex*	*M*	*W*														
ADDU					*IF*	*IF*	*ID*	*Ex*	*M*	*W*													
SW							*IF*	*ID*	*Ex*	*M*	*W*												

It will take 11 cycles for the instructions to completely clear the MIPS pipeline

Waiting for IF to clear

Translate ASM to C

unsigned int r0=0,r1,r3,r4;

```
ADDIU  r1, r0, 0x04
LW r3, 0(r1)
LW r4, 4(r1)
SLL r4, r4, 2
ADDU r4, r4, r3
SW r4, 8(r1)
```

Memory before

0x0c	20
0x08	5
0x04	10
0x00	40

Memory after

0x0c	
0x08	
0x04	
0x00	

r1 = ?

r3 = ?

r4 = ?

Translate ASM to C

```
ADDIU  r1, r0, 0x04
LW r3, 0(r1)
LW r4, 4(r1)
SLL r4, r4, 2
ADDU r4, r4, r3
SW r4, 8(r1)
```

unsigned int r0=0,r1,r3,r4;

r1=r0+4;

r3=(*((unsigned int *)(r1+0)));

r4=(*((unsigned int *)(r1+4)));

r4=r4 << 2;

r4=r4+r3;

(*((unsigned int *)(r1+8)))=r4;

Memory before

0x0c	20
0x08	5
0x04	10
0x00	40

Memory after

0x0c	
0x08	
0x04	
0x00	

r1 = ?

r3 = ?

r4 = ?

Translate ASM to C

```
ADDIU  r1, r0, 0x04
LW r3, 0(r1)
LW r4, 4(r1)
SLL r4, r4, 2
ADDU r4, r4, r3
SW r4, 8(r1)
```

unsigned int r0=0,r1,r3,r4;

r1=4;

r3=(*((unsigned int *)4));

r4=(*((unsigned int *)8));

r4=r4 << 2;

r4=r4+r3;

(*((unsigned int *)12))=r4;

Memory before

0x0c	20
0x08	5
0x04	10
0x00	40

Memory after

0x0c	30
0x08	5
0x04	10
0x00	40

r1 = 4

r3 = 10

r4 = 30