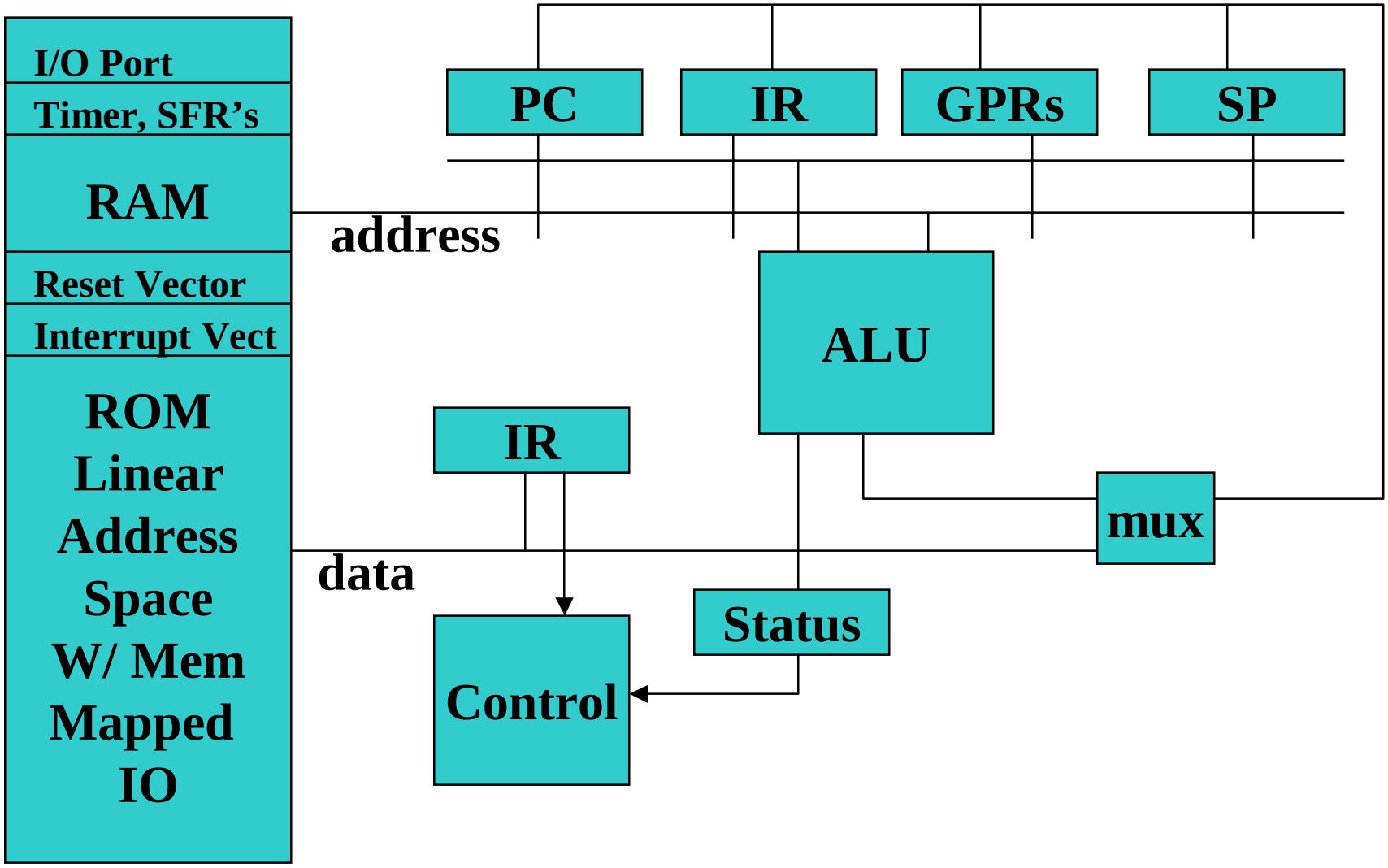
Misc.

- Card Key Access…See Kathleen Goforth
- Mail Archive…working on it…are you getting my messeges?
- Why do we connect the speaker to 5V instead of ground?
- Frequency range … what did you discover?
- Debugger – shows you elapsed simulation time, can set watch variables, etc, etc. Learn more about the debugger!
- Speaker power

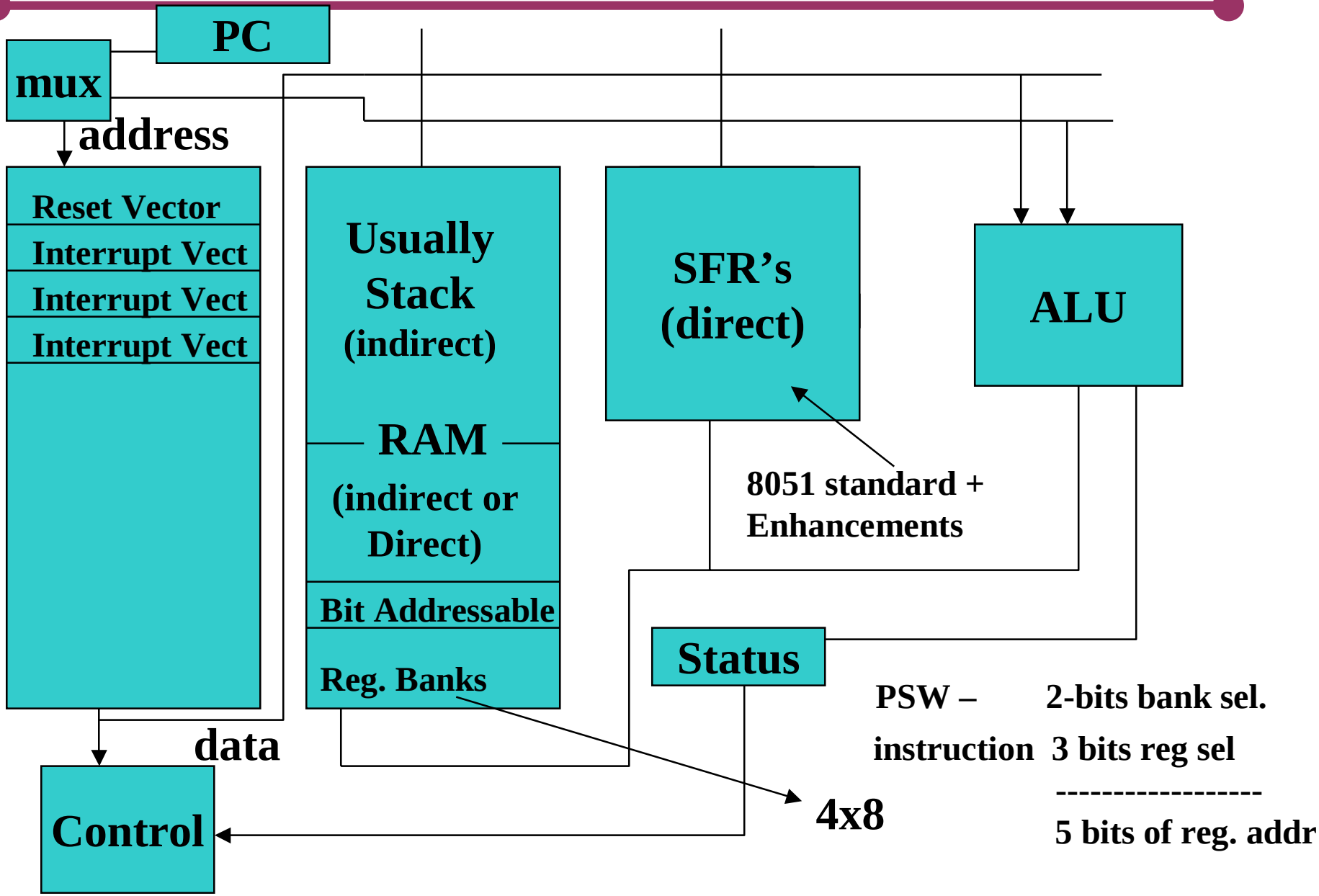
Simple Princeton Architecture



Analysis

- Bottleneck into and out of memory for data and code
- Use of critical 8-bit address space (256) for memory mapped I/O and special function registers (timers and their controllers, interrupt controllers, serial port buffers, stack pointers, PC, etc). For example, the Motorola 6805 processor has only 187 RAM locations.
- But, easy to program and debug. Compiler is simple too.

8051: Modified Harvard Architecture



8051 Memory Architecture

- Advantages
  - Simultaneous access to Program and Data store
  - Register banks great for avoiding context switching on interrupt and for code compression
  - 8-bit address space extended to 256+128 = 384 registers by distinguishing between direct and indirect addressing for upper 128 bytes. Good for code compression
  - Bit addressable great for managing status flags
- Disadvantage
  - A little bit confusing, with potential for errors.

Segments control address space...same in C

```
NAME     example

PROG     SEGMENT  CODE
CONST    SEGMENT  CODE
VAR1     SEGMENT  DATA
BITVAR   SEGMENT  BIT
STACK    SEGMENT  IDATA

         RSEG  BITVAR              ; relocatable segment
flag:    DBIT  1                   ; single bit variables
         RSEG  VAR1                ; relocatable segment
ih:      DS    1                   ; integer i is two bytes
il:      DS    1

         RSEG  STACK               ; relocatable segment
         DS    10H                 ; 16 Bytes

         CSEG  AT   0              ; absolute segment
         JMP   START               ; Execution starts here on reset.

         RSEG  PROG                ; relocatable segment
START:   MOV  SP,#STACK-1          ; first set Stack Pointer
         MOV  PSW,#00              ; use register bank O
         ;rest of main program here
```

what would you add to include an interrupt routine?

```
CSEG AT 0BH
<code>
rti
```

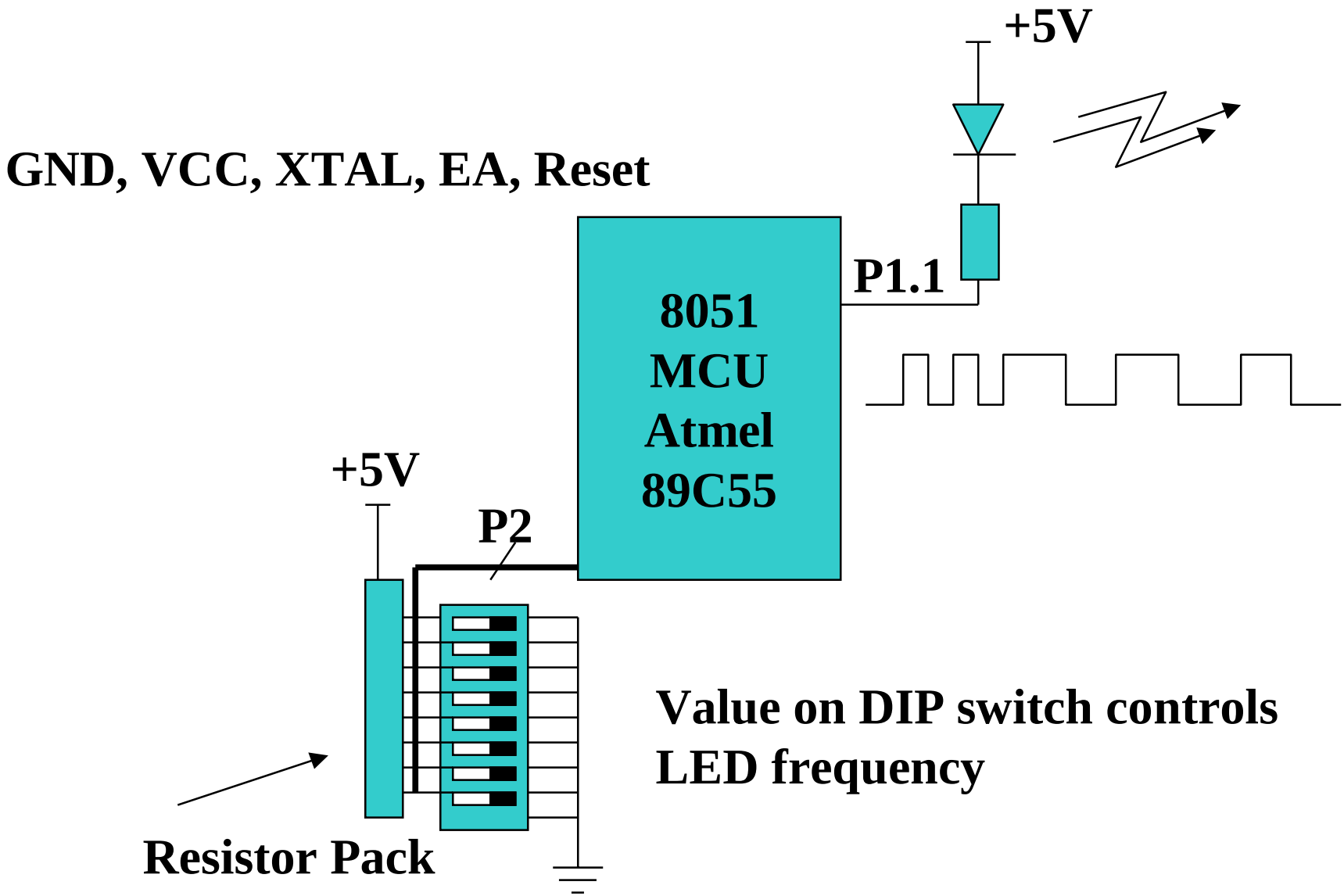
Instruction Execution

- 6 States/Machine Cycle,
  - 2 Osc. Cycles/State = 12 Cycles/Machine Cycle
  - Most instructions are 1 machine cycle, some are 2 or more
- Can make two ROM accesses in on memory cycle (two byte/one cycle instructions, such as ADD A,#10H.
  - ALE – address latch enable, used when referencing external memory which can happen twice per machine cycle.
- Its a Micro-coded CISC processor (sort of an old architecture)
- Interesting features
  - No Zero flag (test accumulator instead)
  - Bit operations, Bit accessible RAM
  - Read Modify Write operations (ports)
  - Register to Register Moves
  - Multiply and Divide operations (many 8-bit MCU's don't have these)
  - Byte and Register Exchange operations
  - Register banks
  - Data pointer registers
  - Addressing Modes (careful when using upper 128 bytes of RAM)
  - BCD oriented instructions

Assembly Programming

- Declare Segments and Segment types
  - Segments define what address space you are in.
  - Assembler converts to machine code, with relocatable segments.
  - Linker perform absolute code location
- Segments
  - DATA -- Internal Data Address Space (0-7F direct or indirect)
  - IDATA -- Indirect Data Address Space (80-FF for stack, arrays)
    - Address is in R0 or R1
  - BIT – Bit addressable RAM space
  - XDATA -- External Data Address Space
  - CODE – Internal or external code space
  - CONST – Internal or external code space
- Example Assembly Program

Last Term's Lab1



Anatomy of an Assembly Program

Look for overflow in C – difficult to do

```
unsigned char i;

void main (void) {
  register unsigned int tmp;
  while (1) {
        P1^= 0x01;
        i = 0;
        do {
           tmp = i;
           i += P2;
        } while (tmp < i);
  }
}
```

```
do {
  i += P2;
} while (!CY)
```

Note i is global and tmp is local. What happens to local variables? How are registers used? What happens in a subroutine call? refer to C51 Manual in Keil (under the books tab, lower left)

Optmized Compiler Result

```
 ; FUNCTION main (BEGIN)
?C0001: XRL     P1,#01H
CLR     A
R     MOV     i,A
?C0005: MOV     R7,i
MOV     R6,#00H
MOV     A,P2
ADD     A,i
MOV     i,A
MOV     R5,A
CLR     C
MOV     A,R7
SUBB    A,R5
MOV     A,R6
SUBB    A,#00H
JC      ?C0005
SJMP    ?C0001
```

Now in Assembly

```
NAME    Lab1_00sp
PUBLIC  il
PUBLIC  ih

PROG    SEGMENT CODE
;CONST  SEGMENT CODE
VAR1    SEGMENT DATA
BITVAR  SEGMENT BIT
STACK   SEGMENT IDATA

        RSEG  BITVAR
flag:   DBIT  1
        RSEG  VAR1
ih:     DS    1
il:     DS    1

        RSEG  STACK
        DS    10H       ; 16 Bytes

        CSEG  AT   0
        USING      0    ; Register-Bank 0
; Execution starts at address 0 on power-up.
        JMP   START
```

```
        RSEG  PROG
; first set Stack Pointer
START:  MOV   SP,#STACK-1
        MOV   PSW,#00   ; SET TO REG BANK O
        CLR    flag     ; just for show
        SETB   flag     ; just for show
LOOP1:  CLR   C         ; Clear carry
        MOV   A,il      ; get low byte
        ADD   A,P2      ; increment
        MOV  il,A
        JNC   LOOP1     ; loop until carry
        INC   ih        ; increment hi byte
        MOV   A,ih      ; check if zero
        JNZ   LOOP1     ;
        XRL   P1,#01H
        SJMP LOOP1
        END
```

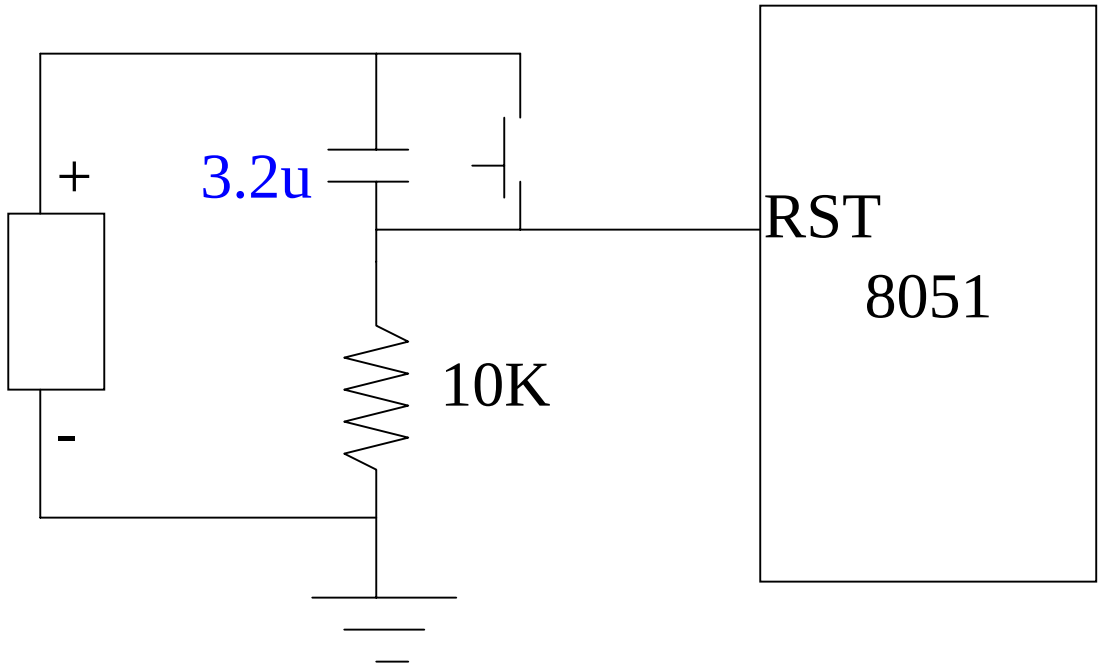
Embedded Hardware

- Microcontrollers
  - Smallest: PIC 8-Pin (8-bit) PIC 8-pin Microcontroller
  - Middle: 6805 (8 bit) Example Flash Based 8051
  - Many 16-bit DSP Microcontrollers
    - HW support for MAC, Filter Algorithms
  - High End: StrongArm (32 bit) Intel
  - Compare to pentium
- External memory
  - Data Address Multiplexing
  - Memory Mapped I/O – talking to external devices
- Typical Devices
  - Resistive Sensors (Strain, Temp, Gas, etc.)
  - Motion sensors (accelerometer)
  - Valve
  - Motor (Stepper, DC, Servo)\
  - Speaker
  - LCD Display
  - LED
  - Latches
  - Gas Sensors

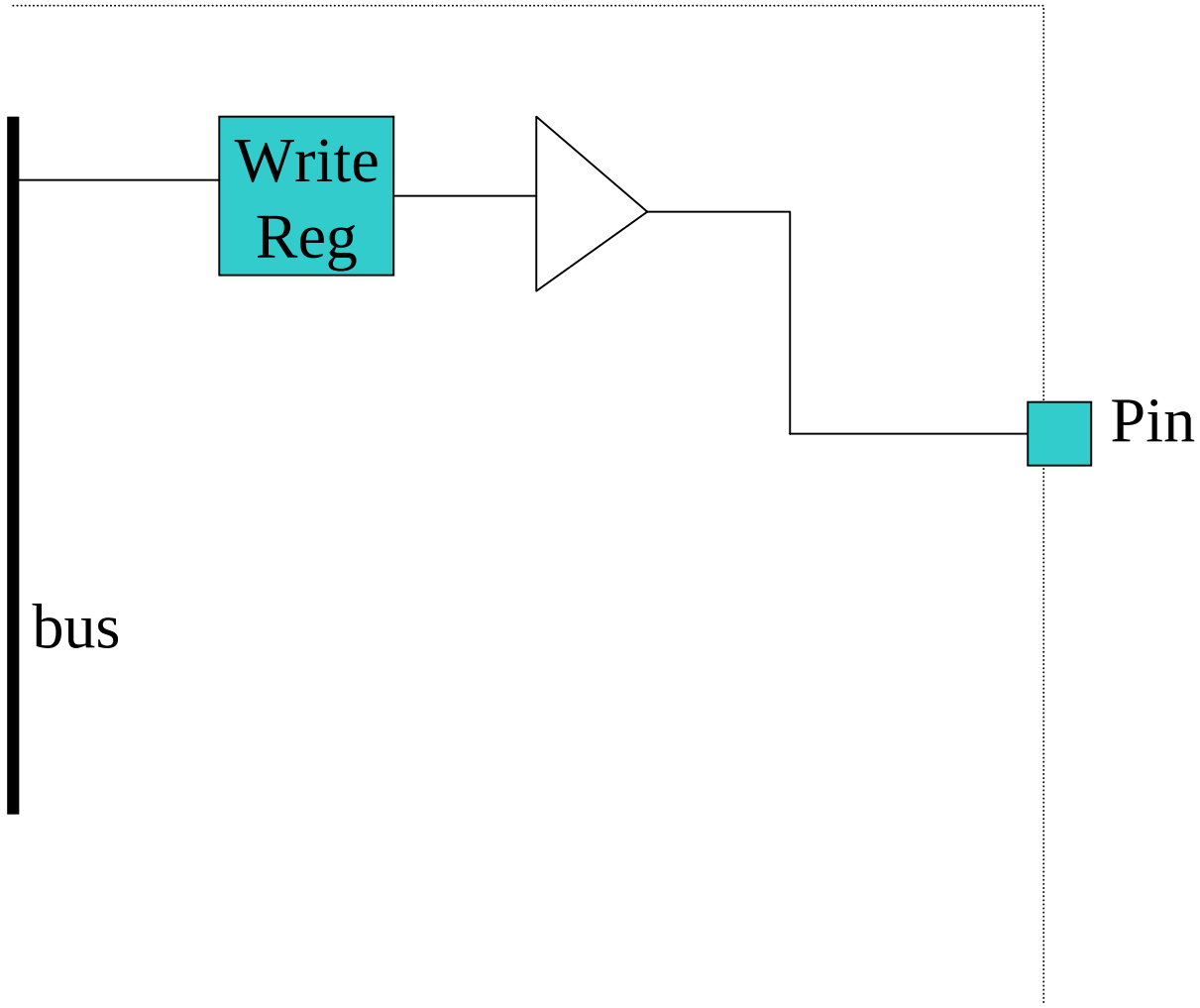
Reset processor 1ms after powerup

- 1ms = 1/32 sec ~ 31ms
- Let R = 10K, so C = .031/10K = 3.1uF

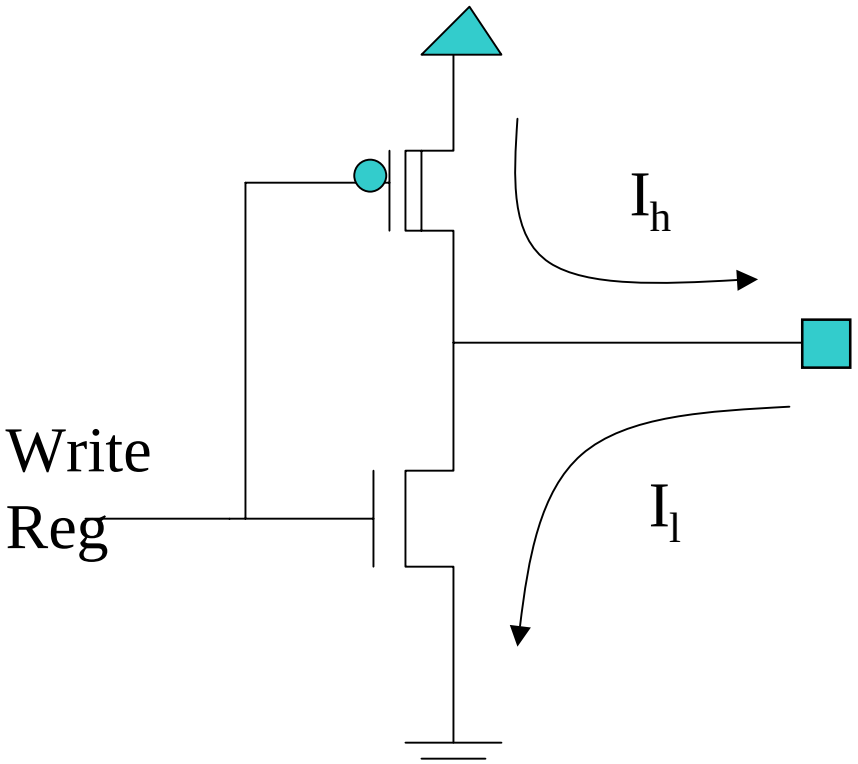
what is the waveform on RST?



An output port



What's Inside the Buffer?



This device always "drives" either high or low.

Current is a function of pin voltage

Never High Impedence 'Z'

Note: this one inverts the signal, but its just an example…