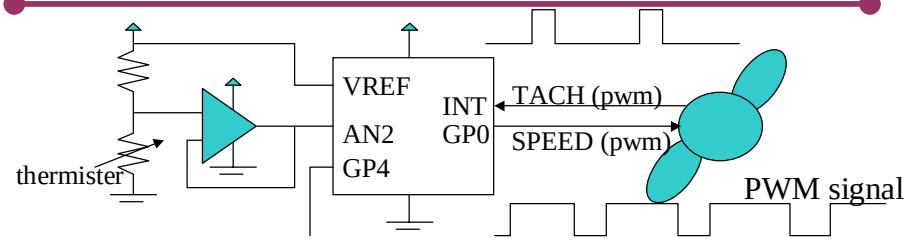
Business Matters

- Book on Reserve. Chapters 1 and 2
- Approx. Quiz and Lecture Schedule is posted (subject to change)
- Lab 1 is posted…we are the only ones using the lab this term…very luxurious
- The Documents
  - Instruction set
  - Architecture overview (8051 Standards)
  - Hardware Description – Applies to all Atmel 8051 Variants (89C55 is like the 89C52 but less flash program memory)
  - 89C55 DataSheet, specifics for our part. Timer2, Electrical Specs, etc.
  - Online: Keil Getting Started Guide, C51 Manual, A51 Manual, BL51 Manual

An Example: Temp Controller w/ 8-Pin PIC MCU



Task: Tachometer (external interrupt)
```
now = getTime();
period = then - now; //overflow?
then = now;
return;
```

Task: TempControl (periodic, soft constraint)
```
if  (Temp > setpoint) Thi++;
if  (Temp < setpoint) Thi--;
if  (period<min || period>max) GP4 = 1;
```

Task: FanPWM (periodic, hard constraint)
```
count++;
if (count == 0) GP0 = 1;
if (count > Thi) GP0 = 0;
return;
```

Task: Main
```
Thi = 0;
setup timer for 1ms interrupt;
setup timer for 100ms interrupt;
while (1) ;
```

Capacity

- Assume:
  - 4 MHz processor @ one instruction/cycle
  - Assume fan runs between 30Hz and 60Hz
  - Assume 256ms period on speed control PWM, with 1ms resolution.
- What percent of the the available cycles are used for the temperature controller?
  - [total instruction in one second] / (4m I/sec)
- How much RAM do you need?
- How much ROM?

Resource Analysis of Temp Controller

Task	ROM	RAM	Instructions/Sec
Tach	~4	2 (period, then)	4 * 60 = 240
FanPWM	~8	1 (count)	8 * 1000 = 8000
TempControl	~10	1 (THI)	10 * 2 = 20

Total Instructions/Sec = 8260, at 4MIPS, thats .2% utilization
Other resources?

- **local variables**
- **stack**

Design Meeting

- Streaming Midi-light Synthesizer
  - Basic Components (Hardware and Software)
  - What should we try first?
  - What do we need to learn about?

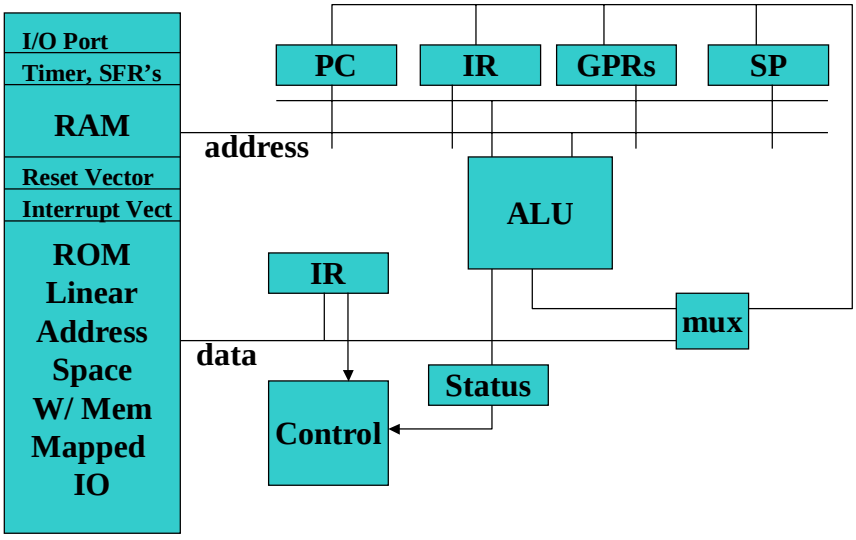
Lab1 Review

- Things to Note:
  - Avoid excessive use of variables (data memory)
  - Avoid excessive use of subroutine calls (stack)
  - Avoid use of floating point (stack, memory, computation)
- To see statistics and assembly result
  - Go to target->options->listing and choose what you want to see in the listing file generated by the compiler (.LST), and in the Linker output file (.M51).
  - Go to the debugger and look at the disassembly window
- Other useful things
  - Code coverage indicator in the debugger
  - Peripherals menu for looking at pins and buffers
  - Debug Functions (scripts for putting patterns on pins during sim)
  - Serial interface simulator for UI debug

Architecture Overview (Start here on Fri)

- Architecture (Harvard v. Princeton)
- Program Memory (ROM)
  - External
  - Internal
- Data Memory (RAM)
  - Direct
  - Indirect
  - SFR's
- Instruction Execution Cycle
  - Risc/Accumulator
  - Microcode
- Structure of an Assembly Language Program
- Bidirectional I/O Ports
- Timers/Counters
  - Modes
- Serial I/O Port
- Interrupt Controller

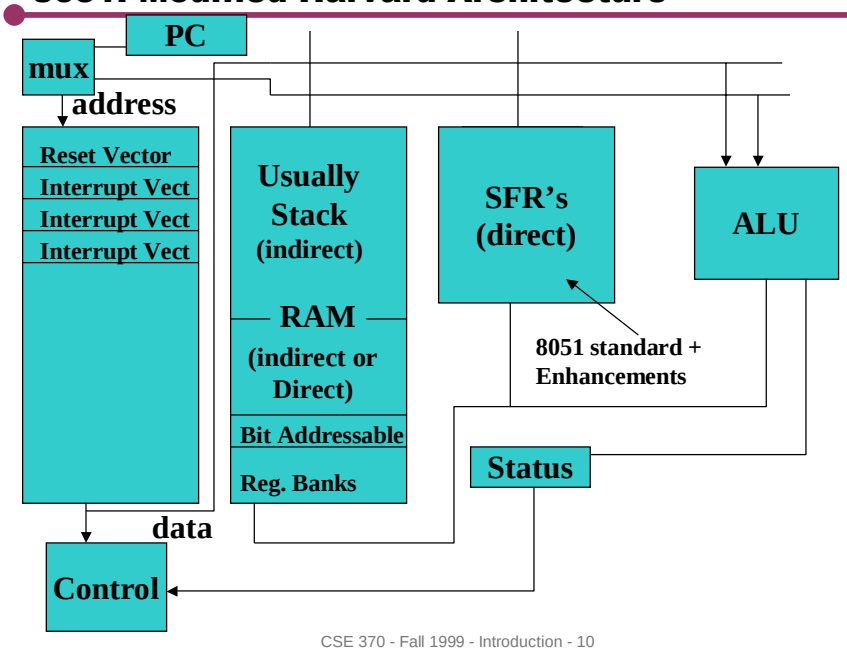
Simple Princeton Architecture



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.