Lecture #11: Final Project

Kunle Olukotun

EE183

February 19, 2003

Lab Issues

- Lab 3 Questions?
  - Memory mapped I/0
  - EXT1
  - Due on Friday @ 6pm
- ASM183
- Modelsim
  - Verilog simulation without synthesis
  - Tutorial and installation instructions EE183 webpage

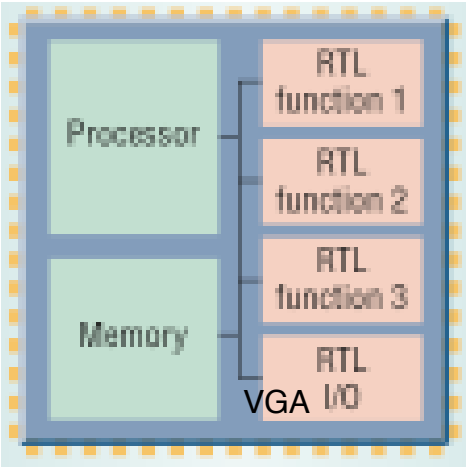
Final Lab

- Start thinking about final lab now!
- Need 1 paragraph description by tonight
- Email to David and I
- Project meetings/feedback by Feb 24

Lab	Pre-Lab Report Due	Demo Due by 5 pm	Final Report Due
1	Jan 17	Jan 24	Jan 27
2	Jan 31	Feb 7	Feb 10
3	Feb 14	Feb 21	Feb 24
4	Feb 28	Mar 14	Mar 17

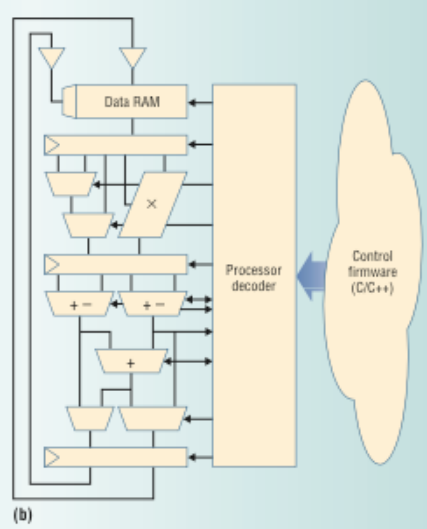
- Bagle Day March 14

Processor Based SOC



- Processors controls hardwired functions
- Memory level communication
  - Memory mapped I/O
  - VGA example
- More software development
- Added flexibility
  - Decreased design time
  - Adapt to changing standards
- Embedded processors
  - MIPS, ARM, PowerPC

Application Specific Instruction-set Processor (ASIP)



- Specialized datapath (FUs)
- Register level communication
- ISA changes
- Advantages
  - Raises design abstraction: C instead of Verilog
  - Eliminates most control logic
  - Added flexibility
- Configurable processors
  - Tensilica
  - ARC
- Final project

Project Examples

- Video game
  - Gamepad and VGA memory mapped I/O
- Keyboard to video character generator
- Fractals with a processor
  - Add multiply unit
- Simple audio processor
  - Interface to audio on XStend board
  - Display on VGA
- Voice disguiser
  - Multiply unit, audio, FIR filter
- DES encryption
- Graphics (line/circle/square/triangle) hardware
- Extra credit for creative applications

Final Project

Part I

Add procedure call and return to your processor
Use register 0 as the link register
Change ASM183 to accommodate your new instructions

Part II

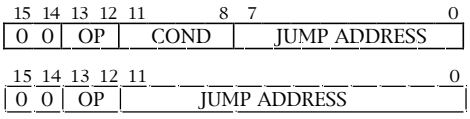
Design an SOC with memory mapped I/O devices or functional units

Design an ASIP by adding a specialized instructions to your 12-bit processor for any application you wish

Control Transfer Instructions

Conditional and unconditional jumps with absolute addresses

Instruction format



- Extra points for making conditional branch PC relative
- Requires assembler changes

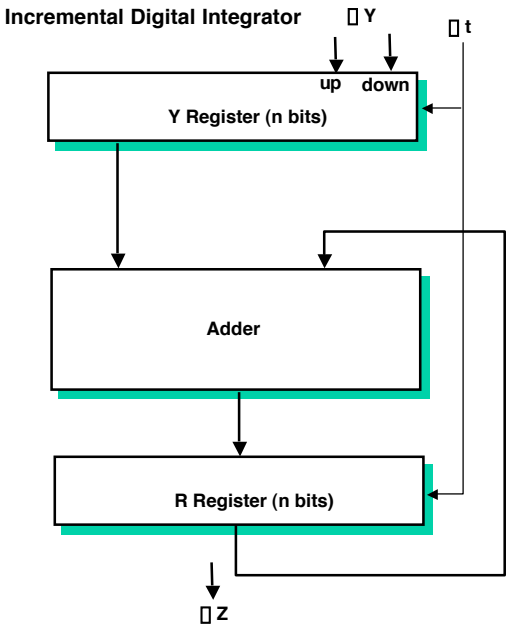
Instructions

OP_{bin}	operation	mnemonic
00	Jump False	`JF.`*cond* *JPC*
01	Jump True	`JT.`*cond* *JPC*
10	Uncond. Jump	`J` *JPC*

$COND_{bin}$	condition	mnemonic
0100	ALU result negative	`.NEG`
0101	ALU result zero	`.ZERO`
0110	ALU carry	`.CARRY`
0111	ALU result negative or zero	`.NEGZERO`
0000	TRUE	`.TRUE`
1000	External Condition	`.EXT`

- Could add up to 8 external conditions

Condition codes are only set by ALU instructions



Carry	Sign Y	ΔZ
1	+	+1
1	-	0
0	+	0
0	-	-1

Original ASM183 Code for Incremental Digital Integrator

```
// R0 = ΔY
// R1 = Y
// R2 = R
// R3 = ΔZ


         ADD              R1, R1, R0
_integ   ADD              R2, R1, R2
         JF.CARRY         _nocarry
         PASSA            R1, R1
         JF.NEGZERO       _plus
         J                _zero
_nocarry PASSA            R1, R1
         JF.NEGZERO       _zero
         J                _neg
_neg     LOADLIT          R3, -1
         J                _end
_pos     LOADLIT          R3, 1
         J                _end
_zero    LOADLIT          R3, 0
_end
```

Define two new instructions

INTEG RC, RA, RB

RC = RA + RB and overflow left in a special register Z

ADDZ RA, RA

RA = RA +1 if Z > 0
RA = RA -1 if Z < 0
RA = RA if Z = 0

New ASM183 Code for Incremental Digital Integrator

```
// R1 = Y
// R2 = R

        ADDZ    R1, R1
        INTEG   R2, R1, R2
```