If you didn't take COMP15111 (ARM assembly code) you should be in IT407

COMP25111: Operating Systems

Lecture 3: Computer Architecture – MU0 Control Signals

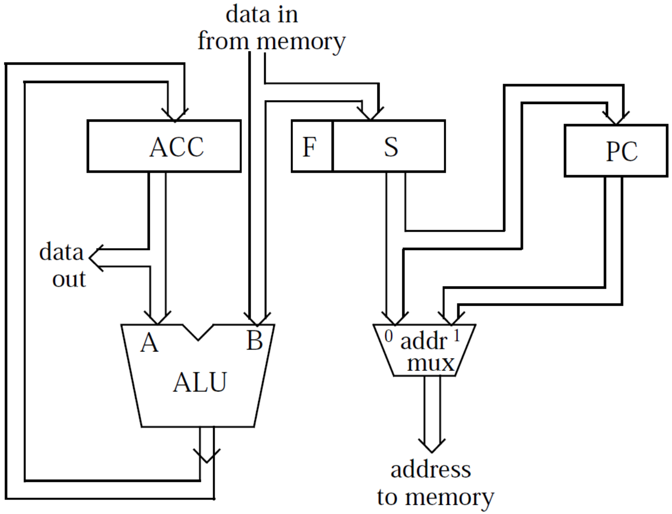
John Gurd

School of Computer Science, University of Manchester

Autumn 2014

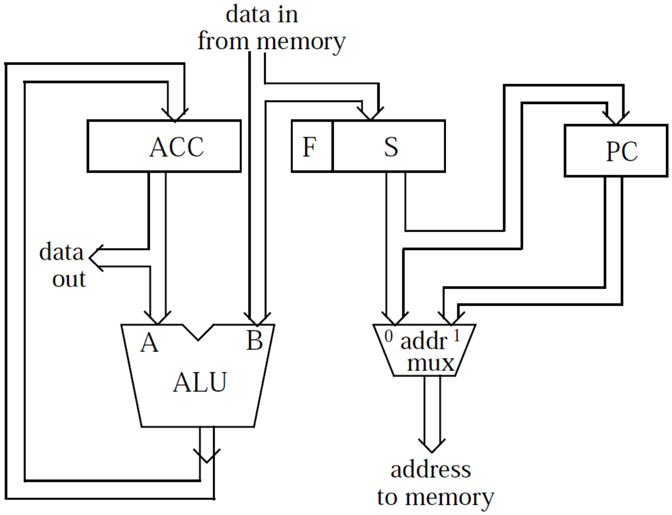
From last time – fetch phase

Shade in the path usage for the fetch phase on the MU0 datapath diagram below:



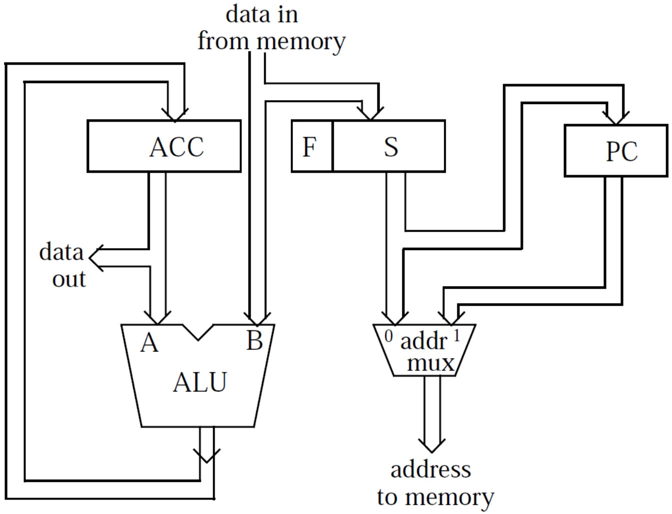
From last time – LDA execute phase

Shade in the path usage for the execute phase for the LDA instructions on the MU0 datapath diagram below:



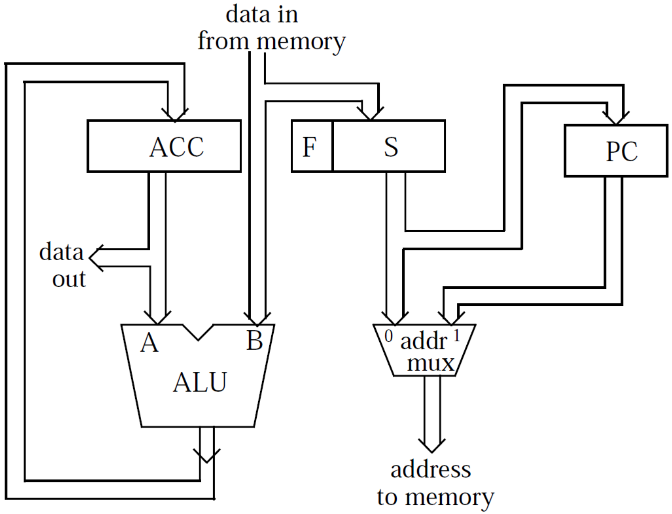
From last time – STA execute phase

Shade in the path usage for the execute phase for the STA instruction on the MU0 datapath diagram below:



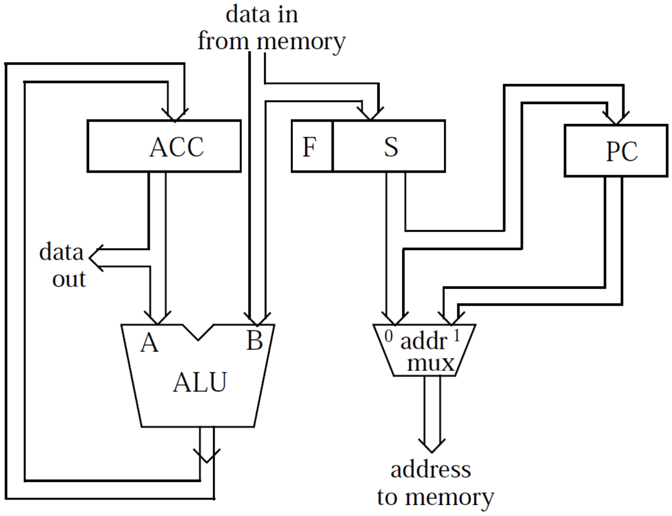
From last time – JMP execute phase

Shade in the path usage for the execute phase for the JMP instruction on the MU0 datapath diagram below:



From last time – ADD execute phase

Shade in the path usage for the execute phase for the ADD instruction on the MU0 datapath diagram below:



Overview & Learning Outcomes

MU0 Control Signals

Introduction to lab 1

Verilog

What Control Signals?

Push: values from registers & devices always available
Control the actions registers & devices perform on inputs

Each register needs an enable, to allow it to be written to:
En_ACC, En_PC, En_IR

Multiplexer needs a signal to select an input

Memory needs actions: Ren (Read Enable) & Wen (Write Enable)

ALU needs actions: add, sub, & byp (bypass)

ALU control signals

We are using three signals (add, sub & byp)

At most one of the three can be set during any phase.

In theory two (encoded) bits would do

Decode either in Control (3 signals) or in ALU (2 signals)

MU0 Control

A “black box”

with these inputs:
external signals: clock reset
from datapaths: F[3:0] N Z
(N & Z come from ACC, needed for JGE & JNE)

and these outputs:
within CPU: En_IR En_PC En_ACC byp add sub addr_Mux
to memory: Ren Wen

Lab 1: What is inside the “black box”?

Must create output signals for each phase of each instruction

These derive from the 6 input bits (F[3:0] N Z)

Some outputs can be “don’t care”
(may simplify the logic circuit)

e.g. ALU action during a JMP

(indicate by “X” instead of “1” or “0”)

Never make register enables or Wen don’t care!

Testing a real chip design

Code in a Hardware Description Language
Then test it (or even synthesise it)

e.g. Verilog language, using Cadence tools.
(beyond the scope of this course)

Verilog is C-like

Control signals are just variables

Verilog code for control state

```
// state: 0 = Reset; 1 = Fetch ; 2 = Execute
always @ (posedge clk)
begin
  if (reset == 1)
    state = 0 ;
  else if (state == 2) & (F == 7)
    state = 2 ; // stp instruction
  else if (state == 2)
    state = 1 ; // Ex->Fetch
  else
    state = state + 1 ; // Reset->Fetch->Ex
end
```

Verilog for Execute Phase

```
case (F)
 0: begin // signals for lda execution
   En_ACC = 1; En_PC = 0; En_IR = 0;
   Ren = 1; Wen = 0; addr_Mux = 0;
   add = ’bx; sub = ’bx; byp = 1;
 end
 1: // similarly for sta
 // etc.
 default:  // undefined instructions
endcase
```

Summary of key points

Building MU0 control signals (Lab 1)
- "Black-box" unit
- Inputs: op-code, conditions, ...
- Outputs: 0/1/X
- Registers: enable write
- Memory & ALU: actions
- Multiplexer: select input

HDL – test, simulate, synthesize

Next lecture: all together for OS concepts

Your Questions

Glossary

Hardware Description Language (HDL)
clock
reset
enable
don’t care

Reading

http://www.cs.man.ac.uk/~pjj/cs1001/mu0_lab.html

http://www.cs.man.ac.uk/~pjj/cs1001/arch/index.html

COMP12111 lecture handouts for “Processors” via
http://www.cs.manchester.ac.uk/ugt/COMP12111/