Computer Architecture Chapter 4 Class 1 Key Topics and Terms

4.1 Introduction

Processor Hardware Described in this Chapter:

MIPS-32 subset implementing

Integer only, no floating point

1) Memory Reference: load word (lw) and store word (sw)

No half-word or byte loads/stores

No atomic operations

2) Arithmetic/Logic: add, sub, and, or, slt

No multiplication, division or shifts

3) Flow of Control: branch equal (beq) and jump (j)

No branch not equal

Jump (j) implementation will be delayed until later sections

Hardware Varieties:

Single Cycle (4.1, 4.3-4.4):

Complete execution of each instruction before moving on

Pipelined (4.5-4.8):

Overlap execution of instructions

Hardware Functions Needed to Implement Instruction Subset (see Figure 4.1):

1) Increment Program Counter (PC) by 4 to move sequentially through code

2) A read-only unit to access memory to *fetch* instructions

Normally, this unit is reading the level 1 (L1) instruction cache

A 32-bit Address input specifies where to get the 32-bit Instruction output

3) A *register file* containing thirty one 32-bit registers

A 32-bit Data input and a 5-bit Register # input specify destination data and destination register number to be written

Two 5-bit Register # inputs specify the two source registers to be read

Notes: a) Two or three of the Register # inputs may have the same values
b) If either (or both) source Register # is the same as the destination Register #, then the old value is output before the new value overwrites the register

4) An *arithmetic and logic unit* (ALU)

Load/store instructions: adds the immediate field of Instruction to one of the source register values from the register file to produce a data address

Arithmetic/Logic instructions: does add, sub, and, or, slt using two source register values

Flow of control instructions: does register value compare on two source register values to determine if equal (beq instruction)

5) A read/write unit to access memory to load/store data

Normally, this unit is reading/writing the level 1 (L1) data cache

A 32-bit Address input specifies where to write the 32-bit Data input or to get the 32-bit unit output

6) An adder sum PC+4 and the immediate field of Instruction to create the next instruction address for a beq instruction if the register values are equal

Notes: a) There are separate adders to generate PC+4 and branch address rather than using ALU such that the ALU does not have to do more than one operation per instruction
b) Adding 4 to PC and then adding the immediate value when branching makes no sense in a single-cycle machine, but it will make sense for the pipelined machine

Control Hardware (see Figure 4.2):

The opcode/funct bits of `Instruction` determine:

1) which of several possible inputs are supplied to hardware units

A multiplexor (MUX) is used to select inputs

2) exactly which function each hardware unit does

Control signals (bold italic below) are said to be

Asserted - condition is true

Deasserted - condition is false

Program Counter (PC):

A 32-bit MUX selects next value input to PC as either

a) PC+4+`Imm` if instruction is beq (***Branch*** asserted) and two register values are equal (***Zero*** asserted)

b) PC+4 otherwise

Register File:

A 32-bit MUX selects `Data` input value as either

a) The 32-bit output value of the ALU (arithmetic/logic instructions)

b) The 32-bit output value of the Data Memory Unit (load word)

Note: this is a don't care for store word or branch equal (or jump)

A control signal ***RegWrite*** determines whether the destination register actually gets written

a) ***RegWrite*** is asserted for load word or arithmetic/logic instructions

b) ***RegWrite*** is deasserted for store word or branch equal (or jump)

Note: the register file always supplies two source register values, but one or both may be nonsense for a given instruction and are ignored by the remaining hardware

ALU:

A 32-bit MUX selects between two possible values for the second ALU input

a) The second register file output (arithmetic/logic instructions and branch equal instruction)

b) The immediate field of Instruction (load word or store word)

Note: this is a don't care for jump

A 4-bit control signal ***ALU operation*** determines which of several possible functions the ALU performs

a) add: load word, store word, add

b) sub: branch equal, sub

c) and, or, slt (for respective arithmetic/logic instructions)

Note: this is a don't care for jump

Data Memory Unit:

A control signal ***MemWrite*** is asserted if the Data input should be written to memory at location specified by the Address input

A control signal ***MemRead*** is asserted if a value should be read from memory at location specified by the Address input

Notes: a) if both ***MemWrite*** and ***Memread*** are both deasserted no operation is performed by the Data Memory Unit
b) ***MemWrite*** and ***Memread*** are never simultaneously asserted
c) Unlike the register file which always reads two values (even if one or both are not used), reading memory from a random address and ignoring the returned value is energy inefficient, can cause delays and possibly cause an error (invalid memory addresses)

4.2 Logic Design Conventions

This section will not be covered