ECE337 / CS341, Fall 2005
Introduction to Computer Architecture and Organization
Instructor: *Victor Manuel Murray Herrera*

Homework # 5 Solutions

Instructions:

- Date Assigned: 11/30/05, 5:30 PM
- Due back: 12/06/05, 8:00 AM

Chapter 12:

12.1 What general roles are performed by CPU registers?

User-visible registers: These enable the machine- or assembly language programmer to minimize main-memory references by optimizing use of registers. **Control and status registers:** These are used by the control unit to control the operation of the CPU and by privileged, operating system programs to control the execution of programs.

12.2 What categories of data are commonly supported by user-visible registers?

General purpose; Data; Address; Condition codes

12.3 What is the function of condition codes?

Condition codes are bits set by the CPU hardware as the result of operations. For example, an arithmetic operation may produce a positive, negative, zero, or overflow result. In addition to the result itself being stored in a register or memory, a condition code is also set. The code may subsequently be tested as part of a conditional branch operation.

12.4 What is a program status word?

All CPU designs include a register or set of registers, often known as the *program status word* (PSW), that contain status information. The PSW typically contains condition codes plus other status information.

12.5 Why is a two-stage instruction pipeline unlikely to cut the instruction cycle time in half, compared with the use of no pipeline?

(**1**) The execution time will generally be longer than the fetch time. Execution will involve reading and storing operands and the performance of some operation. Thus, the fetch stage may have to wait for some time before it can empty its buffer. (**2**) A conditional branch instruction makes the address of the next instruction to be fetched unknown. Thus, the fetch stage must wait until it receives the next instruction address from the execute stage. The execute stage may then have to wait while the next instruction is fetched.

12.6 List and briefly explain various ways in which an instruction pipeline can deal with conditional branch instructions.

Multiple streams: A brute-force approach is to replicate the initial portions of the pipeline and allow the pipeline to fetch both instructions, making use of two streams. **Prefetch branch target:** When a conditional branch is recognized, the target of the branch is prefetched, in addition to the instruction following the branch. This target is then saved until the branch instruction is executed. If the branch is taken, the target has already been prefetched. **Loop buffer:** A loop buffer is a small, very-high-speed memory maintained by the instruction fetch stage of the pipeline and containing the *n* most recently fetched instructions, in sequence. If a branch is to be taken, the hardware first checks whether the branch target is within the buffer. If so, the next instruction is fetched from the buffer. **Branch prediction:** A prediction is made whether a conditional branch will be taken when executed, and subsequent instructions are fetched accordingly. **Delayed branch:** It is possible to improve pipeline performance by automatically rearranging instructions within a program, so that branch instructions occur later than actually desired.

12.7 How is history bits used for branch prediction?

One or more bits that reflect the recent history of the instruction can be associated with each conditional branch instruction. These bits are referred to as a taken/not taken switch that directs the processor to make a particular decision the next time the instruction is encountered.

Chapter 13:

13.1 What are some typical distinguishing characteristics of RISC organization?

(1) a limited instruction set with a fixed format, (2) a large number of registers or the use of a compiler that optimizes register usage, and (3) an emphasis on optimizing the instruction pipeline.

13.2 Briefly explain the two basic approaches used to minimize register-memory operations on RISC machines.

Two basic approaches are possible, one based on software and the other on hardware. The software approach is to rely on the compiler to maximize register usage. The compiler will attempt to allocate registers to those variables that will be used the most in a given time period. This approach requires the use of sophisticated program-analysis algorithms. The hardware approach is simply to use more registers so that more variables can be held in registers for longer periods of time.

13.3 If a circular register buffer is used to handle local variables for nested procedures, describe two approaches for handling global variables.

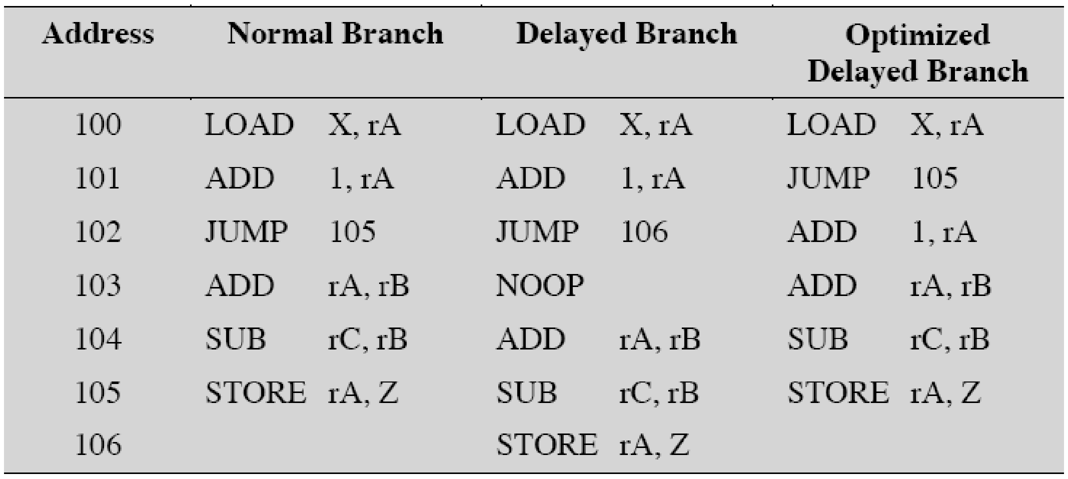
(1) Variables declared as global in an HLL can be assigned memory locations by the compiler, and all machine instructions that reference these variables will use memory-reference operands. **(2)** Incorporate a set of global registers in the processor. These registers would be fixed in number and available to all procedures

13.4 What are some typical characteristics of a RISC instruction set architecture?

One instruction per cycle. Register-to-register operations. Simple addressing modes. Simple instruction formats.

13.5*** What is a delayed branch? And an optimized delayed branch?

Delayed branch, a way of increasing the efficiency of the pipeline, makes use of a branch that does not take effect until after execution of the following instruction.



Address	Normal Branch		Delayed Branch		Optimized Delayed Branch	
100	LOAD	X, rA	LOAD	X, rA	LOAD	X, rA
101	ADD	1, rA	ADD	1, rA	JUMP	105
102	JUMP	105	JUMP	106	ADD	1, rA
103	ADD	rA, rB	NOOP		ADD	rA, rB
104	SUB	rC, rB	ADD	rA, rB	SUB	rC, rB
105	STORE	rA, Z	SUB	rC, rB	STORE	rA, Z
106			STORE	rA, Z		

<u>Chapter 16:</u>

16.1 Explain the distinction between the written sequence and the time sequence of an instruction.

The operation of a computer, in executing a program, consists of a sequence of instruction cycles, with one machine instruction per cycle. This sequence of instruction cycles is not necessarily the same as the **written sequence** of instructions that make up the program, because of the existence of branching instructions. The actual execution of instructions follows a **time sequence** of instructions.

16.2 What is the relationship between instructions and micro-operations?

A micro-operation is an elementary CPU operation, performed during one clock pulse. An instruction consists of a sequence of micro-operations.

16.3 What is the overall function of a processor's control unit?

The control unit of a processor performs two tasks: (1) It causes the processor to execute micro-operations in the proper sequence, determined by the program being executed, and (2) it generates the control signals that cause each micro-operation to be executed.

16.4 Outline a three-step process that leads to a characterization of the control unit.

1. Define the basic elements of the processor. **2.** Describe the micro-operations that the processor performs. **3.** Determine the functions that the control unit must perform to cause the micro-operations to be performed.

16.5 What basic tasks does a control unit perform?

Sequencing: The control unit causes the processor to step through a series of micro-operations in the proper sequence, based on the program being executed. **Execution:** The control unit causes each micro-operation to be performed.

16.6 Provide a typical list of the inputs and outputs of a control unit.

The **inputs** are: **Clock:** This is how the control unit "keeps time." The control unit causes one micro-operation (or a set of simultaneous micro-operations) to be performed for each clock pulse. This is sometimes referred to as the processor cycle time, or the clock cycle time. **Instruction register:** The opcode of the current instruction is used to determine which micro-operations to perform during the execute cycle. **Flags:** These are needed by the control unit to determine the status of the processor and the outcome of previous ALU operations. **Control signals from control bus:** The control bus portion of the system bus provides signals to the control unit, such as interrupt signals and acknowledgments. The **outputs** are: **Control signals within the processor:** These are two types: those that cause data to be moved from one register to another, and those that activate specific ALU functions. **Control signals to control bus:** These are also of two types: control signals to memory, and control signals to the I/O modules.

16.7 List three types of control signals.

(1) Those that activate an ALU function. **(2)** those that activate a data path. **(3)** Those that are signals on the external system bus or other external interface.

16.8 Briefly explain what is meant by a hardwired implementation of a control unit.

In a hardwired implementation, the control unit is essentially a combinatorial circuit. Its input logic signals are transformed into a set of output logic signals, which are the control signals.

Chapter 17:

17.1 What is the difference between a hardwired implementation and a microprogrammed implementation of a control unit?

A **hardwired control unit** is a combinatorial circuit, in which input logic signals are transformed into a set of output logic signals that function as the control signals. In a **microprogrammed control unit**, the logic is specified by a microprogram. A microprogram consists of a sequence of instructions in a microprogramming language. These are very simple instructions that specify micro-operations.

17.2 How is a horizontal microinstruction interpreted?

1. To execute a microinstruction, turn on all the control lines indicated by a 1 bit; leave off all control lines indicated by a 0 bit. The resulting control signals will cause one or more micro-operations to be performed. **2.** If the condition indicated by the condition bits is false, execute the next microinstruction in sequence. **3.** If the condition indicated by the condition bits is true, the next microinstruction to be executed is indicated in the address field.

17.3 What is the purpose of a control memory?

The control memory contains the set of microinstructions that define the functionality of the control unit.

17.4 What is the typical sequence in the execution of a horizontal microinstruction?

The microinstructions in each routine are to be executed sequentially. Each routine ends with a branch or jump instruction indicating where to go next.

17.5 What is the difference between horizontal and vertical instructions?

In a **horizontal microinstruction** every bit in the control field attaches to a control line. In a **vertical microinstruction**, a code is used for each action to be performed and the decoder translates this code into individual control signals.

17.6 What are the basic tasks performed by a microprogrammed control unit?

Microinstruction sequencing: Get the next microinstruction from the control memory. **Microinstruction execution:** Generate the control signals needed to execute the microinstruction.

17.7 What is the difference between packed and unpacked microinstructions?

The degree of packing relates to the degree of identification between a given control task and specific microinstruction bits. As the bits become more **packed**, a given number of bits contains more information. An unpacked microinstruction has no coding beyond assignment of individual functions to individual bits.

17.8 What is the difference between hard and soft microprogramming?

Hard microprograms are generally fixed and committed to read-only memory. **Soft microprograms** are more changeable and are suggestive of user microprogramming.

17.9 What is the difference between functional and resource encoding?

Two approaches can be taken to organizing the encoded microinstruction into fields: functional and resource. The **functional encoding** method identifies functions within the machine and designates fields by function type. For example, if various sources can be used for transferring data to the accumulator, one field can be designated for this purpose, with each code specifying a different source. **Resource encoding** views the machine as consisting of a set of independent resources and devotes one field to each (e.g., I/O, memory, ALU).

17.10 List some common applications of microprogramming.

Realization of computers. Emulation. Operating system support. Realization of special-purpose devices. High-level language support. Microdiagnostics. User Tailoring.

Problems

1) 16.2 (in both editions: 6th and 7th)

LOAD AC:	t_1:	MAR ← (IR(address))	C_8
	t_2:	MBR← Memory	C_5,C_R
	t_3:	AC ← (MBR)	C_{10}

STORE AC	t_1:	MAR $\leftarrow$ (IR(address))	C_8
	t_2:	MBR $\leftarrow$ (AC)	C_{11}
	t_3:	Memory $\leftarrow$ (MBR)	C_{12},C_W
ADD AC	t_1:	MAR $\leftarrow$ (IR(address))	C_8
	t_2:	MBR $\leftarrow$ Memory	C_5,C_R
	t_3:	AC $\leftarrow$ (AC) + (MBR)	C_{ALU},C_7,C_9

Note: There must be a delay between the activation of C_8 and C_9, and one or more control signals must be sent to the ALU. All of this would be done during one or more clock pulses, depending on control unit design.

AND AC	t_1:	MAR $\leftarrow$ (IR(address))	C_8
	t_2:	MBR $\leftarrow$ Memory	C_5,C_R
	t_3:	AC $\leftarrow$ (AC) AND (MBR)	C_{ALU},C_7,C_9
JUMP	t_1:	PC $\leftarrow$ IR(address)	C_3
JUMP if AC= 0		Test AC and activate C_3 if AC = 0	
Complement AC	t_1:	AC $\leftarrow$ ($\overline{AC}$)	C_{ALU},C_7,C_9

2) Suppose that you are working in a research laboratory. The head of the project ask you your opinion to buy a computer, but the doubt is if to buy a CISC processor or and RISC processor. What things do you need to choose a CISC or a RISC? Which one will you buy?

This problem is very subjective. Here you need to think in the most important characteristics about the RISCs and CISCs processor.

The principal reason, for me, to choose one or the other one is about the tasks that that computer will do. If that computer will be used to prove new algorithms, software, programs developed by the researches, an important point of decision is to think that the operating system would not be a problem, one should be free to use all the I/O devices for example. Another important thing is that if the researcher will do his own software, it is probably that he will program in a low level language, in assembler, so with a RISC processor we can perform faster operations but with the cost of the size of the final program.

Now, if the idea is basically to develop improvements of new “powerful” programs, based on Java for example, it is probably that a High Level Language will be used, so a CISC processor appears like a better option.

Which processor one needs to buy? The answer will be based on the specific tasks that the computer will do.