ICS 233
Computer Architecture & Assembly Language

Lecture 3

ICS 233
Computer Architecture & Assembly Language

RISC Processors

Lecture Outline

- Instruction Set Architectures
- RISC Design Methodology
- RISC Characteristics

INSTRUCTION SET ARCHITECTURES

- Two Major Types
  - **Complex Instruction Set Computers (CISCs)**

    Goal of the CISC architecture is to match more closely the operations used in programming languages and to provide instructions that facilitate compact programs and conserve memory.
  - **Reduced Instruction Set Computers (RISCs)**

    Goal of the RISC architecture is high throughput and fast execution.

REDUCED INSTRUCTION SET COMPUTER (RISC)

❑RISC Design Methodology

- **Focuses on reducing the number and complexity of instructions in the machine**
- **A number of strategies employed by RISC designers to exploit caching, pipelining, super-scalarity**

RISC Characteristics

❑Important Characteristics

- **One instruction per cycle**
  - One instruction is issued and executed per cycle.
- **Fixed Instruction length**
  - All instructions are of same length usually 1 word (32 bits)
- **Only Load & Store instructions access memory**
  - Access to operands in memory is limited to two operations, Load & Store.
  - All other operations require the operands to be in registers.

RISC Characteristics

- **Simplified Addressing modes**
  - RISC machines usually limit themselves to only few addressing modes :

    Register, Immediate, Register Indirect or displacement, PC Relative

    In register indirect addressing, register specifies the base and immediate constant contained within the instruction specifies the displacement or offset
- **Limited and simple instruction set**
  - Fewer and simpler operations

RISC Characteristics

- **Large number of general purpose registers**
  - This allows the most frequently accessed operands to be kept in registers and to minimize register-memory operations.
  - Since most operand references are to local scalars, the obvious approach is to store these in registers, with perhaps a few registers reserved for global variables

Register Windows

- **Use of Register windows for speeding up Procedure calls and returns.**
- **A typical procedure employs only a few passed parameters and local variables.**
- **The depth of procedure activation fluctuates within a relatively narrow range.**
- **To exploit these properties, multiple sets of registers are used, each assigned to a different procedure.**
- **A procedure call automatically switches the CPU to use a different fixed-size window of registers, rather than saving registers in memory.**
- **Windows for adjacent procedures are overlapped to allow parameter passing.**

Register Windows

- **The window is divided into three fixed-size areas** :
- **Parameter registers hold parameters passed down from the procedure that called the current procedure and the results to be passed back up.**
- **Local registers are used for local variables, as assigned by the compiler.**
- **Temporary registers are used to exchange parameters and results with next lower level (procedure called by current procedure).**
- **The temporary registers at one level are physically the same as the parameter registers at the next lower level.**
- **This overlap permits parameters to be passed without the actual movement of data.**

RISC Characteristics

- **Use of Compiler technology to optimize register usage**

  - This software approach relies 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.

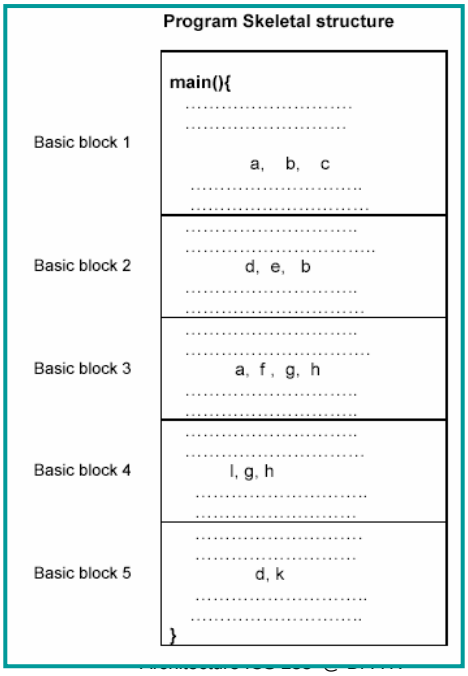
Question

Given a RISC processor with a register file consisting of 32 32-bit registers labeled as R0, R1, R2, ……,R31.

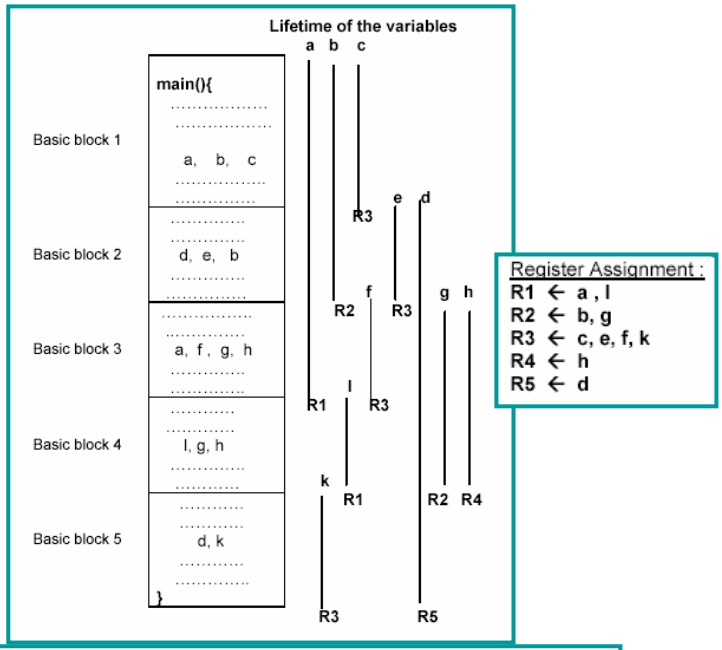
Consider the following program skeletal structure which indicates basic blocks in the program in which the variables a, b, c, d, e, f, g, h, k ,l are used.

It is required to map these program variables to minimum number of registers in the register file in order to achieve register optimization.

Determine the minimum number of registers required to hold these variables by computing the lifetimes of the variables across the basic blocks and assigning these variables to particular registers in the register file.



13



Minimum number of registers required to hold the variables = 5

14

RISC Characteristics

- **Pipelined Instruction Processing Unit**
- **Emphasis on optimizing the instruction pipeline**
  - **Superpipelining**

RISC Characteristics

- **Prefetch and Speculative Execution**

Prefetching

- The advantages of prefetching become even greater in RISC designs.
- **When all instructions are one word long, it becomes more feasible to examine instructions as they enter the pipeline to see if they involve operand access or branching.**
- **If they do, the operand or branch target can be prefetched, resulting effectively in zero execution time for the branch instruction or operand access.**

RISC Characteristics

➢ Speculative Execution

- Having the branch target available, and having several functional units that can execute multiple instructions in parallel, some processors begin execution at the target address in advance of knowing whether the condition has been met, and discard the results if it has not.

  This is known as **speculative execution.**

RISC Characteristics

➢ Delayed Loads and Branches

- **In certain RISC architectures, loads, stores, branch instructions require more than a single clock cycle to execute.**
- **In the case of loads & stores, this delay occurs due to the time required to access memory and in the case of branches, it occurs because of the delay in accessing the instruction stored at branch address.**
- **The concept of delayed loads and branches acknowledges that since they are bound to take more than a single clock cycle to execute, the processor is allowed to execute the instruction following the load or branch while it completes.**
- **This means that the compiler should strive to place an instruction after the delayed instruction that does not depend on its result.**
- **Failing that, a NOP instruction must be located in this position, which is called a delay slot.**

Instruction Set Architecture (ISA)

- **Critical Interface between hardware and software**
- **An ISA includes the following …**
  - Instructions and Instruction Formats
  - Data Types, Encodings, and Representations
  - Programmable Storage: Registers and Memory
  - Addressing Modes: to address Instructions and Data
  - Handling Exceptional Conditions (like division by zero)
- **Examples** **(Versions)** **First Introduced in**
  - Intel (8086, 80386, Pentium, ...) 1978
  - MIPS (MIPS I, II, III, IV, V) 1986
  - PowerPC (601, 604, …) 1993

Instructions

- **Instructions are the language of the machine**
- **We will study the MIPS instruction set architecture**
  - Known as **Reduced Instruction Set Computer (RISC)**
  - Elegant and relatively simple design
  - Similar to RISC architectures developed in mid-1980's and 90's
  - Very popular, used in many products
    - Silicon Graphics, ATI, Cisco, Sony, etc.
  - Comes next in sales after Intel IA-32 processors
    - Almost 100 million MIPS processors sold in 2002 (and increasing)
- **Alternative design: Intel IA-32**
  - Known as **Complex Instruction Set Computer (CISC)**