Memory Access

Memory Access Modes

COMP375 Computer Architecture and Organization

Steps in Address Mapping

- Programmer selects a variable name.
- Compiler allocates space and selects a relative address.
- Linker combines object files and updates references to relative address to be program addresses.
- Operating system allocates memory for the program and copies it into RAM.

Relocatable Address Adjustment

<table>
<thead>
<tr>
<th>Relocatable Object Files</th>
<th>Executable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 100</td>
<td>0 0 100</td>
</tr>
<tr>
<td>0 50 150</td>
<td>0 101 151</td>
</tr>
<tr>
<td>0 152</td>
<td>0 152 302</td>
</tr>
</tbody>
</table>

Adjusting Addresses

- When the linker combines object files to create an executable, it goes through the program and adjusts every address.
- When a program is executed, it is loaded into memory. The hardware adjusts each program relative address to a hardware address.
Program Memory Types

- Global variables
  - Data segment
- Local variables and parameters
  - Stack
- Dynamic variables
  - Heap
- Constants
  - Data segment or instruction segment
- Instructions
  - Instruction segment

Program Memory Organization

Effective Address

- The effective address (or logical address or virtual address) is the program relative address.
- The hardware maps the effective address to the physical address.
- Different programs in different physical addresses may have the same effective address.
- Effective addresses can be the result of address calculations.

Where is the data?

- The operands for an instruction can be in RAM, a register or in the instruction.
- The instruction specifies where the operand is located.
- There are several ways the operand’s address can be specified.
- Different addressing modes have been created for different program situations.
### Addressing modes

- immediate
- register
- memory direct
- register indirect
- register indirect with offset
- memory indirect
- register + offset memory indirect
- displacement

### Immediate

- The data is part of the instruction.
- Immediate data items are read-only.
- There is usually a size limit.

### Register

- The data is in a CPU register.
- The instruction *might* indicate which register

### Memory Direct

- The data is in memory.
- The instruction contains the address of the memory location.
Register Indirect

- The address of the data is in a CPU register.
- Useful if the address is calculated.

Memory Indirect

- A memory location contains the address of the data.
- Useful for pointers.

Register Indirect with Offset

- The address of the data is the sum of the instruction offset field and a register value.
- Useful when addressing an array.

Register Offset & Memory Indirect

- The sum of the instruction offset field and a register value gives the location of the address in memory.
- Useful when addressing ref parameters.
Displacement

- The address of the data is the sum of the instruction offset field and the program counter.
- Used for short jumps

Stack Addressing

- The same as Register Indirect with Offset using the stack pointer register.
- Useful when addressing local variables or parameters.

Function Calls

- To call a function or method, the program counter is pushed on the stack and then the program counter is loaded with the address of the function.
- This puts the address of the instruction after the function call on the stack.
- To return the return address is popped from the stack and loaded into the program counter.

What are the Instructions?

- Table of opcodes for different instructions including add, subtract, multiply, divide, load, store, jump, and jump equal.
- Instruction formats for load and store, add, subtract, multiply, and divide with different register combinations.
- Binary representation of instruction formats and opcodes for each instruction category.
Notes on the Example Architecture

• This is an example of a “Load/Store” architecture. Only the load and store instructions access memory.
• Why is there a one bit unused field in the arithmetic instructions?
• Why is the opcode always the left most bits?
• The format of the jump instructions was not shown. What might be a good format?

Instruction Cycle

• Fetch the instruction from the memory address in the Program Counter register
• Increment the Program Counter
• Decode the type of instruction
• Fetch the operands
• Execute the instruction
• Store the results

Basic Processor Components

• **Program Counter** – contains the address of the next instruction to execute.
• **Arithmetic Logic Unit** – logic to perform arithmetic and logical functions
• **User registers** – hold data
• **Memory Address Register** – contains the address to be copied to or from RAM
• **Memory Buffer Register** – contains data copied to or from RAM.

Instruction Fetch
Memory Access

Instruction Fetch

Increment Program Counter

Increment Program Counter

Decode Instruction
Jump Instruction

- Consider an arithmetic instruction followed by a jump instruction.
- The arithmetic instruction sets bits in the status register.

Execution Stage of Instruction 1

Result Save Stage of Instruction 1

Instruction 2 Fetch
Instruction 2 Fetch

Increment Program Counter

Increment Program Counter

Decode Instruction
Goals for Instruction Set

• The purpose of the machine language is to run the applications.
• Compilers should be able to translate high level languages into machine language.
• Run fast

RISC vs. CISC

• Complex Instruction Set Computers
  – many instructions
  – some instructions can perform complex operations.
  – each instruction may take several cycles
• Reduced Instruction Set Computers
  – fewer instructions
  – many RISC machines run 1 instruction / cycle
CISC Theory

- The machine language should be designed to be as close as possible to the application requirements.
- Machine instructions should support high level language constructs.

RISC Theory

- Many complex instructions are rarely executed. Most programs use only a small set of instructions.
- The few important instructions should run as fast as possible.
- If you have to do something complex, use several fast simple instructions.
- Registers are much faster than RAM.

PDP-8 Instruction Set

- The PDP-8 was a popular mini computer made by DEC in the 1970’s.
- The 12 bit fixed length instructions had a 3 bit op code, an address field and two addressing mode bits.
- There were 7 instructions with the normal format.
- One op code was for zero operand instructions. The remaining bits of the instruction specified what to do.