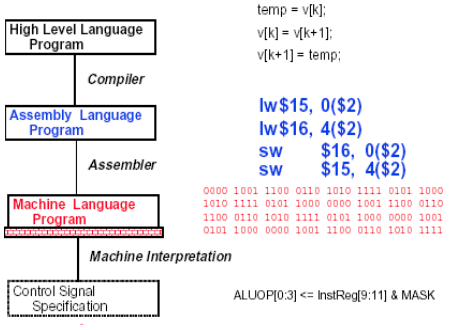
ECE 3401 Lecture 21

Instruction Set Architecture

Overview

- Computer architecture
- Operand addressing
  - Addressing architecture
  - Addressing modes
- Elementary instructions
  - Data transfer instructions
  - Data manipulation instructions
    - Floating point computations
  - Program control instructions
    - Program interrupt and exceptions

Levels of Representation



Computer Architecture

- Instruction set architecture
  - A set of hardware-implemented instructions, the symbolic name and the binary code format of each instruction
- Organization
  - Structures such as datapath, control units, memories, and the busses that interconnect them
- Hardware
  - The logic, the electronic technology employed, the various physical design aspects of the computer

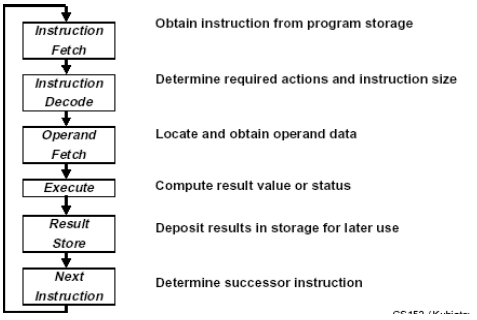
Example ISAs (Instruction Set Architectures)

- RISC (Reduced Instruction Set Computer)
  - Digital Alpha
  - Sun Sparc
  - MIPS RX000
  - IBM PowerPC
  - HP PA/RISC
- CISC (Complex Instruction Set Computer)
  - Intel x86
  - Motorola 68000
  - DEC VAX
- VLIW (Very Large Instruction Word)
  - Intel Itanium

Instruction Set Architecture

- A processor is specified completely by its ***instruction set architecture (ISA)***
- Each ISA will have a variety of instructions and instruction formats, which will be interpreted by the processor's control unit and executed in the processor's datapath
- An instruction represents the smallest indivisible unit of computation. It is a string of bits grouped into different numbers and size of ***substrings*** (***field***s)
  - ***Operation code (opcode):*** *the operation to be performed*
  - ***Address field:*** *where we can find the operands needed for that operation*
  - ***Mode field:*** how to derive the data's effective address from the information given in the address field
  - ***Other fields:*** *constant immediate operand or shift*

Computer Operation Cycle



Register Set

- Programmer accessible registers (R_0 to R_7 in previous multi-cycle computer)
- Other registers
  - Registers in the register file accessible only to microprograms (R_8 to R_{15})
  - Instruction registers (IR)
  - Program counter (*PC*)
  - Pipeline registers
  - Processor status register (*PSR*: CVNZ state)
  - Stack pointer (*SP*)

Overview

- Computer architecture
- Operand addressing
  - Addressing architecture
  - Addressing modes
- Elementary instructions
  - Data transfer instructions
  - Data manipulation instructions
    - Floating point computations
  - Program control instructions
    - Program interrupt and exceptions

Operand Addressing

- Operand: register value, memory content, or immediate
- *Explicit address*: address field in the instruction
- *Implied address*: the location of operand is specified by the opcode or other operand address

Three Address Instructions

- Example: X=(A+B)(C+D)
- Operands are in memory address symbolized by the letters A,B,C,D, result stored memory address of X

ADD T1, A, B	M[T1] ← M[A] + M[B]
ADD T2, C, D	M[T2] ← M[C] + M[D]
MUX X, T1, T2	M[X] ← M[T1] X M[T2]

OR

ADD R1, A, B	R1 ← M[A] + M[B]
ADD R2, C, D	R2 ← M[C] + M[D]
MUX X, R1, R2	M[X]← R1 X R2

- +: Short program, 3 instructions
- -: Binary coded instruction require more bits to specify three addresses

Two Address Instructions

- The first operand address also serves as the implied address for the result

MOVE T1, A	M[T1] ← M[A]
ADD T1, B	M[T1] ← M[T1] + M[B]
MOVE X, C	M[X] ← M[C]
ADD X, D	M[X] ← M[X] + M[D]
MUX X, T1	M[X] ← M[X] X M[T1]

- 5 instructions

One Address Instructions

- Implied address: a register called *an accumulator ACC* for one operand and the result, *single-accumulator architecture*

```
LD    A    ACC ← M[A]
ADD   B    ACC ← ACC + M[B]
ST    X    M[X] ← ACC
LD    C    ACC ← M[C]
ADD   D    ACC ← ACC + M[D]
MUX   X    ACC ← ACC X M[X]
ST    X    M[X] ← ACC
```

7 instructions

- All operations are between the ACC register and a memory operand

Zero Address Instructions

- Use stack (FILO):
  - ADD TOS ← TOS + TOS_{-1}
  - PUSH X TOS ← M[X]
  - POP X M[X] ← TOS

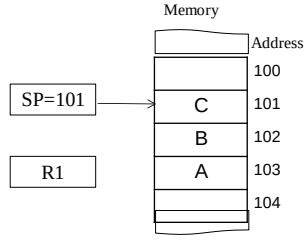
```
PUSH A    TOS ← M[A]
PUSH B    TOS ← M[B]
ADD       TOS ← TOS + TOS-1
PUSH C    TOS ← M[C]
PUSH D    TOS ← M[D]
ADD       TOS ← TOS + TOS-1
MUX       TOS ← TOS X TOS-1
POP  X    M[X] ← TOS
```

8 instructions

- Data manipulation operations: between the stack elements
- Transfer operations: between the stack and the memory

Stack Instructions

- Push:
  - SP ← SP-1; TOS ← R1
- Pop:
  - R1 ← TOS; SP ← SP +1



Addressing Architecture

- Defines:
  - Restriction on the number of memory addresses in instructions
  - Number of operands
- Two kinds of addressing architecture:
  - Memory-to-memory architecture
    - Only one register - PC
    - All operands from memory, and results to memory
    - Many memory accesses
  - Register-to-register (load/store) architecture
    - Restrict only one memory address to load/store types, all other operations are between registers

```
LD   R1, A        R1 ← M[A]
LD   R2, B        R2 ← M[B]
ADD  R3, R1, R2   R3 ← R1 + R2
LD   R1, C        R1 ← M[C]
LD   R2, D        R2 ← M[D]
ADD  R1, R1, R2   R1 ← R1 + R2
MUL  R1, R1, R3   R1 ← R1 X R3
ST   X, R1        M[X] ← R1
```


Addressing Modes

- Address field: contains the information needed to determine the location of the operands and the result of an operation
- Addressing mode: specifies how to interpret the information within this address field, how to compute the **actual** or **effective** address of the data needed.
- Availability of a variety of addressing modes lets programmers write more efficient code

Addressing Modes

- Implied mode - implied in the opcode, such as stack, accumulator
- Immediate mode (operand) - a = 0x0801234
- Register mode - a=R[b]
- Register-indirect mode – a =M[R[b]]
- Direct addressing mode - a = M[0x0013df8]
- Indirect Addressing mode - a= M[M[0x0013df8]]
- PC-relative addressing – branch etc. (offset + PC)
- Indexed addressing - a=b[1]

Example

ADRS or NBR=500

PC=250

R1=400

ACC

Addressing mode	Symbolic conversion	Register transfer	Effective address of	Content of ACC
Immediate	LDA #NBR	ACC←NBR	-	500
Register	LDA R1	ACC←R1	-	400
Register-indirect	LDA (R1)	ACC←M[R1]	400	700
Direct	LDA ADRS	ACC←M[ADRS]	500	800
Indirect	LDA [ADRS]	ACC←M[M[ADRS]]	800	300
Relative	LDA $ADRS	ACC←M[ADRS+PC]	750	600
Index	LDA ADRS(R1)	ACC←M[ADRS+R1]	900	200

	Opcode	Mode
250		
251	Next instruction	
252		
400	700	
500	800	
750	600	
800	300	
900	200	

19

Overview

- Computer architecture
- Operand addressing
  - Addressing architecture
  - Addressing modes
- Elementary instructions
  - Data transfer instructions
  - Data manipulation instructions
    - Floating point computations
  - Program control instructions
    - Program interrupt and exceptions

Instruction Set Architecture

	RISC (reduced instruction set computers)	CISC (complex instruction set computers)
Memory access	restricted to load/store instructions, and data manipulation instructions are register-to-register	is directly available to most types of instructions
Addressing mode	limited in number	substantial in number
Instruction formats	all of the same length	of different lengths
Instructions	perform elementary operations	perform both elementary and complex operations
Control unit	Hardwired, high throughput and fast execution	Microprogrammed, facilitate compact programs and conserve memory,

21

Data Transfer Instructions

- Data transfer: memory ←→ registers, processor registers ←→ input/output registers, among the processor registers
- Data transfer instructions

Name	Mnemonic
Load	LD
Store	ST
Move	MOVE
Exchange	XCH
Push	PUSH
Pop	POP
Input	IN
Output	OUT

22

I/O

- Input and output (I/O) instructions transfer data between processor registers and I/O devices
  - Ports
- Independent I/O system: address range assigned to memory and I/O ports are independent from each other
- Memory-mapped I/O system: assign a subrange of the memory addresses for addressing I/O ports

23

Data Manipulation Instructions

Arithmetic		Logical and bit manipulation		Shift instructions	
Name	**Mnemonic**	**Name**	**Mnemonic**	**Name**	**Mnemonic**
Increment	INC	Clear	CLR	Logical shift right	SHR
Decrement	DEC	Set	SET	Logical shift left	SHL
Add	ADD	Complement	NOT	Arithmetic shift right	SHRA
Subtract	SUB	AND	AND	Arithmetic shift left	SHRL
Multiply	MUL	OR	OR	Rotate right	ROR
Divide	DIV	Exclusive-OR	XOR	Rotate left	ROL
Add with carry	ADDC	Clear carry	CLRC	Rotate right with carry	RORC
Subtract with borrow	SUBB	Set Carry	SETC	Rotate left with carry	ROLC
Subtract reverse	SUBR	Complement carry	COMC		
Negate	NEG				

24

Floating-Point Computation

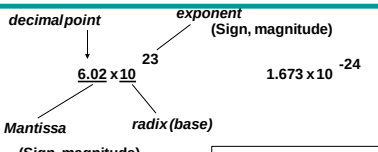
- What can be represented in N bits?

Unsigned	0	to	2^N-1
2s Complement	-2^{N-1}	to	$2^{N-1}-1$
1s Complement	$-2^{N-1}+1$	to	$2^{N-1}-1$
BCD	0	to	$10^{N/4}-1$

- But, what about?
  - very large numbers? 9,349,398,989,787,762,244,859,087,678
  - very small number? 0.00000000000000000000000045691
  - rationals 2/3
  - irrationals $\sqrt{2}$
  - transcendentals e

25

Recall Scientific Notation



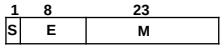
- Issues:
  - Representation, Normal form
  - Range and Precision
  - Arithmetic (+, -, *, /)
  - Rounding
  - Exceptions (e.g., divide by zero, overflow, underflow)
  - Errors
  - Properties (negation, inversion, if A ≠ B then A - B ≠ 0)

IEEE F.P. $\pm 1.M \times 2^{e-127}$

26

Floating-Point Numbers

- **Representation of floating point numbers in IEEE 754 standard:**
  - **single precision**



exponent: binary integer

mantissa: sign + magnitude, normalized binary significand w/ hidden integer bit: 1.M

Biased exponent: actual exponent is e = E – 127 (0 < E < 255)

$N = (-1)^S \times 2^{E-127} \times (1.M)$

0 = 0 00000000 0 . . . 0 **-1.5 = 1 01111111 10 . . . 0**

- **Exponent field (E):**
  E=0 reserved for zero (with fraction M=0), and denormalized #s (M ≠ 0)
  E=255 reserved for ±∞ (with fraction M=0), and NaN (M ≠ 0)
- **Magnitude of numbers that can be represented is in the range: (with E in [1, 254]):**

$2^{-126} \times (1.8 \times 10^{-38}) \sim 2^{127} \times (2-2^{-23}) \times (3.4 \times 10^{38})$

27