Machine Language

Last updated 6/14/23

These slides introduce machine language

Machine Language

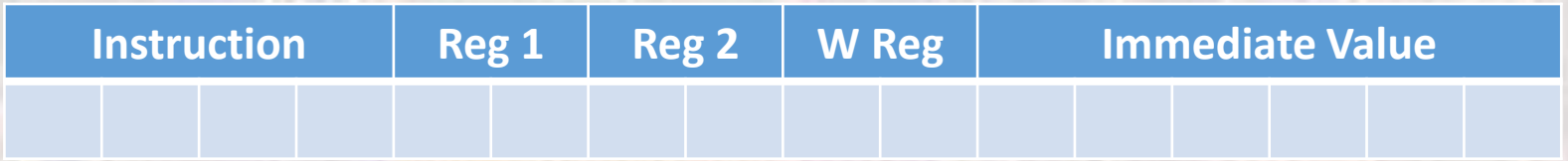
- Machine Language
  - The representation of assembly language in a binary format
    - Computers are made of hardware and only understand 1s and 0s
  - This is the values stored in program memory
    - Identical processors can share a machine language
    - Non-identical processors require unique machine languages

Machine Language

- Pretend Processor Architecture
  - Harvard Architecture
  - RISC – Load/Store Instruction Set
  - 16 bit instruction words
  - 4 – 8 bit data registers available for executing instructions (A-B-C-D)
  - Support for 16 instructions
  - 3 – memory based instructions

Machine Language

- Instruction Encoding
  - Convert assembly language to machine language
  - 16 bits program words



or	0000
and	0001
nor	0010
nand	0011
add	0100
sub	0101
slt	0110
ld	1000
st	1001
ldi	1100
bre	1110

00 – A
01 – B
10 – C
11 – D

signed Hex
0x20 to 0x1F

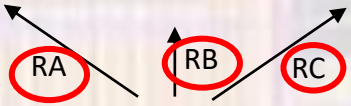
100000 to 011111
-32 to 31

Machine Language

- Instruction Encoding – reg/reg
  - sub Wreg ← Reg1 - Reg2
  - sub RA, RB, RC RC ← RA - RB

Instruction				Reg 1		Reg 2		W Reg		Immediate Value					
0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0

or	0000
and	0001
nor	0010
nand	0011
add	0100
sub	0101
slt	0110
ld	1000
st	1001
ldi	1100
bre	1110



00 – A
01 – B
10 – C
11 – D

Technically these are
don't care
but
we will always code them as 0s

Machine Language

- Instruction Encoding – Mem
  - ld — Wreg ← MEM(Reg1)
  - ld RB, RC — RC ← MEM(RB)

Instruction				Reg 1		Reg 2		W Reg		Immediate Value					
1	0	0	0	0	1	x	x	1	0	0	0	0	0	0	0

or	0000
and	0001
nor	0010
nand	0011
add	0100
sub	0101
slt	0110
ld	1000
st	1001
ldi	1100
bre	1110

RB

RC

00 – A
01 – B
10 – C
11 – D

Technically these are
don't care
but
we will always code them as 0s

Machine Language

- Instruction Encoding – Mem
  - st — MEM(Reg1) ← Reg2
  - st RB, RC — MEM(RB) ← RC

Instruction				Reg 1		Reg 2		W Reg		Immediate Value					
1	0	0	1	0	1	1	0	x	x	0	0	0	0	0	0

or	0000
and	0001
nor	0010
nand	0011
add	0100
sub	0101
slt	0110
ld	1000
st	1001
ldi	1100
bre	1110

00 – A
01 – B
10 – C
11 – D

Technically these are
don't care
but
we will always code them as 0s

Machine Language

- Instruction Encoding – Immediate
  - ldi — Wreg ← "imm value"
  - ldi RD, 0x24 — RD ← 0x24

Instruction				Reg 1		Reg 2		W Reg		Immediate Value					
1	1	0	0	x	x	x	x	1	1	1	0	0	1	0	0

or	0000
and	0001
nor	0010
nand	0011
add	0100
sub	0101
slt	0110
ld	1000
st	1001
ldi	1100
bre	1110

RD

00 – A
01 – B
10 – C
11 – D

0x24

Machine Language

- Instruction Encoding – Branch
  - bre — PC ← PC + offset if REG = 0
  - bre RD, 0x24 — PC ← PC + 0x24 if RD = 0

Instruction				Reg 1		Reg 2		W Reg		offset Value					
1	1	1	0	1	1	x	x	x	x	1	0	0	1	0	0

or	0000
and	0001
nor	0010
nand	0011
add	0100
sub	0101
slt	0110
ld	1000
st	1001
ldi	1100
bre	1110

RD

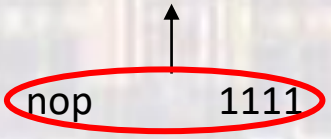
00 – A
01 – B
10 – C
11 – D

0x24

Machine Language

- Special function
  - nop — do nothing
  - nop

Instruction				Reg 1		Reg 2		W Reg		Immediate Value					
1	1	1	1	x	x	x	x	x	x	0	0	0	0	0	0



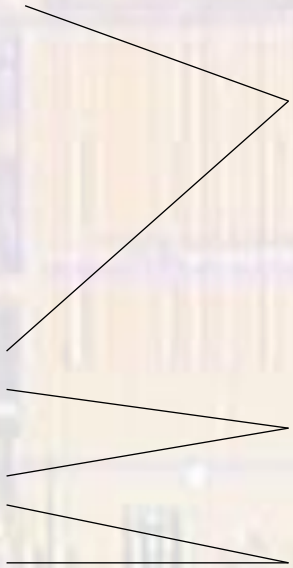
00 – A
01 – B
10 – C
11 – D

Machine Language

- Instruction Format

Instruction				Reg 1		Reg 2		W Reg		Immediate Value					

or	0000
and	0001
nor	0010
nand	0011
add	0100
sub	0101
slt	0110
ld	1000
st	1001
ldi	1100
bre	1110
nop	1111



Reg/Reg

MEM

IMM

Wreg ← Reg1 fn Reg2

Wreg ← MEM(Reg1)

MEM(Reg1) ← Reg2

Wreg ← “imm value”

PC ← PC + “offset” if REG = 0