

CoreWars

Ref: Handout, Web Links

CompOrg Fall 2000 - Core Wars

1

CoreWars

- Developed by A.K. Dewdney 1984
- Two programs battle to the death.
 - inspired by early worm/antiworm folklore.
- Programs are written in a language similar to assembly language.
 - There is no real processor – it is simulated.

CompOrg Fall 2000 - Core Wars

2

Redcode and Mars

- The language (instruction set) is called RedCode.
- The simulator is “Memory Array Redcode Simulator” or Mars.
- There are versions of Mars for Windows, Dos and Unix.

CompOrg Fall 2000 - Core Wars

3

Playing Field

- There is a *circular* memory array, originally it had 8000 locations.
 - Modern tournaments use different size playing fields.
- Two programs are loaded in memory and executed one instruction at a time (alternating turns).
- The last program *alive* wins!

CompOrg Fall 2000 - Core Wars

4

Winning

- When one program attempts to execute an invalid instruction – it is killed.
- The general idea is to insert faulty instructions in to the other program.
 - there is no way to know where the other program is.
 - the other program is trying to do the same thing.

CompOrg Fall 2000 - Core Wars

5

Redcode

- No registers – everything is in memory.
- No absolute addresses – everything is relative to the PC.
- A number of *addressing modes*
- Only 8 instructions
 - and a data declaration statement.

CompOrg Fall 2000 - Core Wars

6

Redcode Instructions

arithmetic: **ADD** and **SUB**

data movement: **MOV**

transfer control: **JMP JMZ JMG DJZ CMP**

Data declaration: **DAT**

CompOrg Fall 2000 - Core Wars

7

MOV instruction

- **MOV A B**: move from **A** to **B**
- **A** and **B** can be specified using the addressing modes:
 - direct (PC relative address)
 - immediate (preceded by a #)
 - indirect (preceded by a @)

CompOrg Fall 2000 - Core Wars

8

MOV examples

Move from address PC+3 to PC+100:

MOV 3 100

Move the number 17 to address PC+10:

MOV #17 10

Move the number found at the address found
in location PC+4 to the location PC+8:

MOV @4 8

CompOrg Fall 2000 - Core Wars

9

ADD and SUB

- **ADD A B**
 - add **A** to **B**
 - store the result in **B**
- Same addressing modes as **MOV**
- **SUB** works the same way.

CompOrg Fall 2000 - Core Wars

10

ADD & SUB Examples

Add 3 to the value stored at address PC+7 and store the sum in address PC+7:

ADD #3 7

Subtract one from the number stored at the address found in location PC+4. Store the result at PC+4:

SUB #1 @4

CompOrg Fall 2000 - Core Wars

11

Transfer Control (*jump*)

- JMP A:** jump to **A**
- JMZ A B:** jump to **A** if the contents of **B** is zero
- JMG A B:** jump to **A** if the contents of **B** are greater than zero.
- DJZ A B:** subtract 1 from contents of address **B** and jump to **A** if the result is zero.
- CMP A B:** if contents of **A** is not equal to contents of **B** skip the next instruction.

CompOrg Fall 2000 - Core Wars

12

Examples

Jump ahead 10 instructions:

JMP 10

If the number stored at PC+4 is 0, jump back 3 instructions:

JMZ -3 4

Decrement the number found at the address found in location PC+4 and jump ahead 5 instructions if the result is zero:

DJZ 5 @4

CompOrg Fall 2000 - Core Wars

13

Data Statement

DAT B: stores the number **B** in memory.

- Not an instruction – a declaration.
 - see the example program “dwarf”

DAT -1

DAT 0

CompOrg Fall 2000 - Core Wars

14

Machine Language

- Each instruction is encoded as a decimal integer.
 - The simulator deals with these integers.
- The instruction 0 is invalid.
 - put a 0 in the middle of the other program and you will win.

CompOrg Fall 2000 - Core Wars

15

DWARF program

```
DAT -1
ADD #5 -1      Throws "bombs" in every
                5th location in memory
MOV #0 @-2
JMP -2
```

CompOrg Fall 2000 - Core Wars

16

Mars supports symbols

```
bomb    DAT #-1
dwarf   ADD #5, bomb
        MOV bomb, @bomb
        JMP dwarf
        END dwarf
```

CompOrg Fall 2000 - Core Wars

17

IMP program

```
MOV 0 1
```

- Program relocates itself every instruction!
- A moving target.
- Leaves behind a trail of MOV 0 1 instructions.

CompOrg Fall 2000 - Core Wars

18

Gemini: Copies itself to new position and transfers control to the new copy.

DAT	0		<i>pointer to source address</i>
DAT	99		<i>pointer to dest address</i>
MOV	@-2	@-1	<i>copy source to dest.</i>
CMP	-3	#9	<i>all 10 line of the program copied?</i>
JMP	4		<i>yes – done (leave loop)</i>
ADD	#1	-5	<i>no – update src address.</i>
ADD	#1	-5	<i>and dest. address</i>
JMP	-5		<i>and loop again.</i>
MOV	#99	93	<i>new dest. address</i>
JMP	93		<i>jump to new copy</i>
