AT2 Language
Reference Manual

Table of Contents

Introduction

Here's a complete documentation about the AT2 language.
All the information provided is accurate, agreed and reviewed by all the team members.
You can use this document to create your own .aop files.

SYSTEM

Registers

The Virtual Processor is composed of 8 registers of 16 bits each.
From rg0 to rg7 each is usable except rg3 which is reserved to the clock and it's in read only.

ex:

```
mov rg0, 9
draw rg0
// Print the content of rg0 which is 9
```

Lexical Conventions

Comments

A comment can be appended to a statement.
The comment consists of the double slash character (//) followed by the text of the comment.
The comment is terminated by the newline that terminates the statement.

ex:

```
mov rg0, 9
add rg0, 6
// My comment is ignored by the program
draw rg0
```

Labels

Labels are subroutines that contain instructions.
They can be called at any moment during the execution of code by goto and call mnemonics.
To declare them, the syntax is lab name.

ex:

```
draw “Hello world!!”
lab myLabel
        draw “Goodbye world!!”
```

Variables

A variable is a named container for a particular set of bits or type of data. It can be declared anywhere in the code. It needs to have a name and a value : *var name, value*.

ex:

var name, “AT2”
// Set the string AT2 into name variable
Draw **name**

Strings

A string is a sequence of characters, used to represent text. You can declare them using: " ”.

ex:

var name, **“AT2”**
// you declare the string: AT2
draw name
// Will display the content of name
// result: AT2

Characters

A character refers to a single unit of text or symbol. You can declare it using: ‘ ‘.

ex:

var char, **‘A’**
// you declare the character: A
draw char
// Will display the content of char
// result: A

Hexadecimal

The lexical convention for representing hexadecimal numbers in most programming languages, including AT2. To use it, prefix the hexadecimal value with 0x.

ex:

var color, **0x**AF76AB
// you declare the character: 0xAF76AB
draw name
// Will display the content of color
// result**:** 0xAF76AB

Binary

The lexical convention for binary values in the AT2 assembly language is represented by a sequence of digits, where each digit can be 0 or 1. To use it, prefix it with 0b.

ex:

```
var letter, 0b01000001
// you declare the character:  0b01000001 (A)
draw, name
// Will display the content of letter
// result:  A
```

INSTRUCTIONS

General-Purpose Instructions

Data Transfer Instructions

The data transfer instructions move data between memory and the general-purpose and segment registers, and perform operations such as conditional moves, stack access, and data conversion.

AT2 Mnemonic	Description	Example
mov	copy the data immediate value and paste it to another location	mov rg0, rg1
push	push into stack	push rg0
pop	pop last from stack	pop rg0
pusha	push all registers into stack	pusha
popa	pop all registers from stack	popa

Binary Arithmetic Instructions

The binary arithmetic instructions perform basic integer computations on operands in memory or the general-purpose registers.

AT2 Mnemonic	Description	Example	Note
+ Or **add**	addition	**+** rg0, 2 **add** rg0, 2	
- Or **sub**	subtraction	**-** rg0, 2	
/ Or **div**	division	**/** rg0, 2	
***** Or **mul**	multiplication	***** rg0, 2	
% Or **mod**	modulo	**%** rg0, 2	
| Or **or**	bitwise OR	**|** rg0, 2	
& Or **and**	bitwise AND	**&** rg0, 2	
^ Or **xor**	bitwise XOR	**^** rg0, 2	
! Or **not**	bitwise NOT	**!** rg0, 2	
neg	negate the value	**neg** rg0	
++ Or **inc**	increment	**++** rg0	
-- Or **dec**	decrement	**--** rg0	

Comparison Instructions

The comparison instructions ensure that users have a straightforward method, closely resembling higher-level languages, to compare variables.

if	first compare	if (=, rg0, 2)	between if and the first parenthesis one space is required
else	second compare	else	
end	close if and else compare statement	end	

Logical Instructions

The logical instructions perform basic logical operations on their operands.

AT2 Mnemonic	Description	Example
&&	AND	if (&&, rg0, rg1)
||	OR	if (||, rg0, rg1)

^^	XOR	if (^, rg0, rg1)
<	inferior	if (<, rg0, rg1)
>	superior	if (>, rg0, rg1)
<=	inferior or equal	if (<=, rg0, rg1)
>=	superior or equal	if (>=, rg0, rg1)
==	equal	if (=, rg0, rg1)
!=	not equal	if (!=, rg0, rg1)

Shift Instructions

Shift instructions move the bits of a binary number to the left or right within a register or memory location.

AT2 Mnemonic	Description	Example
>> Or **shr**	shift the bits to the right	>> rg0
<< Or **shl**	shift the bits to the left	<< rg0

Control Transfer Instructions

The control transfer instructions control the flow of program execution.

AT2 Mnemonic	Description	Example	Note
call	call subroutine	call label	
goto	go to subroutine	goto label	
ret	return where previous call was use	ret	only when call is use

String Instructions

String instructions perform operations on strings.

AT2 Mnemonic	Description	Example
draw	print the string on the display	draw "Hello, World!"

I/O Instructions

The input/output instructions transfer data between the processor's I/O ports, registers, and memory.

AT2 Mnemonic	Description	Example	Note
get	get external file content	get "filename.aop"	get instructions need to be at the header of file

Operating System Support Instructions

These instructions provide support for interfacing with the operating system.

AT2 Mnemonic	Description	Example
clok	get the current execution time of program (only with register 3)	clok
ngr	exit the program and return control to the operating system	ngr

INDEX