AT2 Language reference manual

Table of contents

Introduction

This document serves as the complete documentation for the AT2 language, its utilization, and features.
All the information provided is accurate and has been agreed upon and reviewed by all the team members.
This document can be used to create your .aop files.

SYSTEM

Registers

The Virtual Processor is composed of 8 registers of 16 bits each.
From rg0 to rg7, all are usable except rg3, which is reserved for the clock and is read-only.

ex:

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

Register 0

This register is commonly used to draw values and to compare statements with if_*. The register 0 will always be before the register 1.

Register 1

This register is commonly used to compare statements with if_*. The register 1 will always be after the register 0.

Register 3

The register3 (rg3) cannot be used as storage for values. He can only give his own values to registers or variables.

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 containing 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:

```
lab start
    mov rg0, 100
    goto myLabel
lab myLabel
    draw
```

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

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

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

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	Note
mov	copy the data immediate value and paste it to another location	mov rg0, 3 mov rg1, rg0	First argument cannot be an immediate value.
push	push into stack	mov rg0, 5 push rg0	We can only push existing registers
pop	pop last from stack	mov rg0, 5 push rg0 pop rg0	We can't pop rg3 or any inexisting registers
pusha	push all registers into stack	pusha	Push all registers in stack except rg3
popa	pop all registers from stack	popa	Pop all registers from stack except rg3

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	mov rg0, 2 **add** rg0, 2	
-	Or	**sub**	subtraction	mov rg0, 4 **-** rg0, 2	
/	Or	**div**	division	mov rg0, 4 **/** rg0, 2	
*****	Or	**mul**	multiplication	mov rg0, 4 ***** rg0, 2	
%	Or	**mod**	modulo	mov rg0, 4 **%** rg0, 2	
|	Or	**or**	bitwise OR	mov rg0, 4 **|** rg0, 2	
&	Or	**and**	bitwise AND	mov rg0, 4 **&** rg0, 2	
^	Or	**xor**	bitwise XOR	mov rg0, 4 **^** rg0, 2	
!	Or	**not**	bitwise NOT	mov rg0, 4 **!** rg0, 2	

Logical instructions

The logical instructions perform basic logical operations on their operands.

AT2 Mnemonic			Description	Example	Note
<	Or	**if_lt**	lower than	if_it	
>	Or	**if_gt**	greater than	if_gt	
<=	Or	**if_lte**	lower than or equal	if_ite	
>=	Or	**if_gte**	greater than or equal	if_gte	
==	Or	**if_eq**	equal	if_eq	
!=	Or	**if_neq**	not equal	if_neq	

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	Note
>> Or **shr**	shift the bits to the right	>> rg0, 2	Right shift operation
<< Or **shl**	shift the bits to the left	<< rg0, 2	Left shift operation

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	Note
draw	print the register 0 content	draw	Use Reg 0

Operating system support instructions

These instructions provide support for interfacing with the operating system.

AT2 Mnemonic	Description	Example	Note
ngr	exit the program and return control to the operating system	ngr	Check no additional arguments

INDEX

Category	AT2 Mnemonic		Description	Page
-	-	sub	subtraction	page 6
!	!	not	bitwise NOT	page 6
*****	*	mul	multiplication	page 6
/	/	div	division	page 6
&	&	and	bitwise AND	page 6
%	%	mod	modulo	page 6
^	^	xor	bitwise XOR	page 6
+	+	add	addition	page 6
	++	inc	increment	page 6
<	<	if_it	inferior	page 7
	<<	shl	shift left	page 7
	<=	if_ite	inferior or equal	page 7
=	==	if_eq	equal	page 7
>	>	if_gt	superior	page 7
	>=	if_gte	superior or equal	page 7

INDEX

Category	AT2 Mnemonic		Description	Page
	>>	shr	shift right	page 7
|	|	or	bitwise OR	page 6
C	call		call label	page 7
D	draw		draw	page 8
G	goto		goto	page 7
M	mov		mov	page 6
N	ngr		ngr	page 8
P	pop		pop	page 6
	popa		popa	page 6
	push		push	page 6
	pusha		pusha	page 6
R	ret		ret	page 7