Sharif University of Technology

Department of Computer Engineering

Compiler Course Project

Fall 1396

Handout 6 - Instruction specification

1 Instructions

Instructions appear in one line of output each and will have the following format:

[OpCode] [Opr1] . . . [Opr$n$]

Where the number of operands (n) is determined for each instruction separately. All of the required instructions in L is listed in table below. If you need any instructions that is not listed here, please inform me.
Note: The last three instructions will be discussed in project's extra class.

Operator	OpCode	# of Operands	Operation
Add	+	3	[Opr3] ← [Opr1] + [Opr2]
Subtract	−	3	[Opr3] ← [Opr1] − [Opr2]
Multiply	*	3	[Opr3] ← [Opr1] [Opr2]
Divide	/	3	[Opr3] ← [Opr1] / [Opr2]
Mod	%	3	[Opr3] ← [Opr1] % [Opr2]
Logical And	&&	3	[Opr3] ← [Opr1] && [Opr2]
Logical Or	||	3	[Opr3] ← [Opr1] || [Opr2]
Binary And	&	3	[Opr3] ← [Opr1] & [Opr2]
Binary Or	|	3	[Opr3] ← [Opr1] | [Opr2]
Binary Xor	^	3	[Opr3] ← [Opr1] ^ [Opr2]
Binary Not	~	2	[Opr2] ← ~[Opr1]
Less Than	<	3	[Opr3] ← [Opr1] < [Opr2]
Greater	>	3	[Opr3] ← [Opr1] > [Opr2]
Binary Left	<<	2	[Opr2]← [Opr2] << [Opr1]
Binary Right	>>	2	[Opr2]← [Opr2] >>[Opr1]
Less Than	<=	3	[Opr3] ← [Opr1] <= [Opr2]
Greater Than	>=	3	[Opr3] ← [Opr1] >= [Opr2]
Equal	==	3	[Opr3] ← [Opr1] == [Opr2]
Not Equal	! =	3	[Opr3] ← [Opr1] ! = [Opr2]
Logical Not	!	2	[Opr2] ← ! [Opr1]
Unary Minus	*u−*	2	[Opr2] ← −[Opr1]
Assignment	:=	2	[Opr2] ← [Opr1]
Jump Zero	*jz*	2	if [Opr1]==TRUE then pc ←[Opr2]
Jump	*jmp*	1	pc ← [Opr1]
Write	*wi*	1	{output} ← [Opr1]
Write Float	*wf*	1	{output} ← [Opr1]
Write Text	*wt*	1	{output} ← [Opr1]
Read Integer	*ri*	1	{input} → [Opr1]
Read Float	*rf*	1	{input} → [Opr1]
Read Text	*rt*	1	{input} → [Opr1]
Get	*gmm*	2	*Set Opr2 the address of firt byte of memmory with size Opr1*
Free	*fmm*	2	*Free memory starts at Opr1 with size Opr2*
PC Value	:=*pc*	1	[Opr1] ← pc
SP Value	:=*sp*	1	[Opr1] ← sp
Assign SP	*sp* :=	1	sp ← [Opr1]
Increase SP*	+*sp*	1	sp ← sp + [Opr1]
Decrease SP*	-*sp*	1	sp ← sp − [Opr1]
OverFlow Value	:=*v*	1	[Opr1] ← OverFlow register

1.1 Operands

Each of the operands of the following format:

[Addressing Mode] [Type] [Value]

You have to concatenate their text values in order to obtain the operand. For immediate addressing, value will be the literal (for a character, you will write it's ASCII code). In other kind of addressing, value will be a memory address (integer).

1.2 Addressing Modes

in *L* we will need at most five kind of addressing mode.

Addressing Mode	Text Form
Global Direct	gd_
Global Indirect	gi_
Local Direct	ld_
local Indirect	li_
immediate	im_

1.3 Types

Type	Text Form
Integer	i_
Float	f_
Boolean	b_
String	s_
Char	c_

2 Example

In this section you can see some examples from instructions in text form. the white space between operator and operands can be a single space or tab.

```
rt gd_s_0
wt  gd_s_0
```

```
+   gd_i_12    im_i_5    ld_i_14
wi   im_c_13
*   gi_f_10    im_f _10.5   ld_f_10
```

```
 gmm    im_i_1024    gd_i_1
 wi    gi_i_1
 +    im_i_1    im_i_2    gi_i_1
 wi    gi_i_1
fmm    gd_i_1    im_i_1024
```

```
 *   im_i_2147483647   im_i_2   gd_i_100
 :=v   gd_b_0
 wi   gd_i_0
```