System Class Requirements for Project 1

Project Description:

Project is the implementation of an interpreter that will process Dalvik bytecode. The interpreter is to be written in Javascript and supported by a common browser. The output of the interpreter should execute Javascript code.

A number of Dalvik Op Codes are defined and listed here: http://www.netmite.com/android/mydroid/dalvik/docs/dalvik-bytecode.html

System Requirements:

- 1) The system shall provide the capability to read in a file that contains bytecode data
- 2) The system shall parse the bytecode data
- 3) The system shall map the dalvik syntax to javscript syntax
- 4) The system shall produce an output of results
- 5) A project report must be created that is no more than 5 pages
- 6) A project package must be created that includes instructions for (Setup, Build, Run and Test), Source and Binary code.
- 7) The customer will provide details of acceptance test before final due date
- 8) Project completion date is approximately four weeks from start Sept 23, 2012

Software Requirements:

- 1) FireFox 15.0.X or higher will be the common browser that will run the interpreter code
- 2) A Linux environment will be used to develop, test and perform the user acceptance test
- 3) A web server shall be supported to host the HTML and Interpreter code
- 4) The Web page shall support a file upload mechanism
- 5) A list of Dalvik Op Codes that will be supported is shown below:

Register to register move 0 - 0.04 Umited Not nuncating bits Nove exception Move exception to a register Umited Not nuncating bits Nove exception Nove	Instruction	Description	OpCode	Supported
Move exception Move exception to a register No No No No No No No N	move	Register to register move	01 -0d	
In a value, void	move-result			Limited No truncating bits
Monton-cent	Move-exception			No
Monitral exist	return			
Manifer exist		Move a constant into register dest		
Check cost			1d -1e	No
Instance of Array-length services and the services of the serv		those Olse Oset Franchise Year agent he first	45	No
Aray-length				
New Instance		Set dest to 1 or 0 according to whether sic is type		
New array				
Fill-array Dut Items from data into array at dest 27		see filled-new-array for argument registers?		
Introver Strate Static-get Static-ge			20 20	
goto jump to a different jace within the method 28 - 2a switch execute a switch statement 2b No Packet switch 2b No Sparse switch 2c No omp primitive_type may be float, double, or long 2d if 32-38 If Ifeq if (a == b) goto address 32-38 Iffeq if (a == b) goto address Iffet Iffeg Iffeg Iffeg Iffeg Iffeg Iffeq Iffeg Iffeg Iffeg Iffeg Iffeg<		-	27	
switch execute a switch statement 2b No Sparse switch 2c No Crmp primitine_type may be float, double, or long 2d If If If If If If If If				
Sparse switch	~			
cmp primitive_type may be float, double, or long 2d if eq if (a == b) goto address if-eq if (a == b) goto address if-the iff. iff.ge iff.ge	Packet switch		2b	No
	Sparse switch		2c	No
	стр	primitive_type may be float, double, or long	2d	
If-re		-	32-38	
If-It If-I	lf-eq	If (a == b) goto address		
If-gt				
If-tt	lf-ge			
If-riez	lf-lt			
If-tiz If-gez	lf-eqz	if(src == 0) goto address		
If-gez	If-nez			
If-gitz	lf-ltz			
If-itz	_			
Array-get				
Array-pet Array-put instance Instance-get Instance-get Instance-put Static-get Static-get Static-put Instance Call a method of kind = {\text{virtual, super, direct, static, interface}} The British per = {\text{int, long, byte, float double}} Primitive-cast Get Ge	lf-ltz			
Array-put Instance	Array		44-51	
Instance_get Inst		Primtype = { int, wide, object, boolean, byte, char, short }		
Instance-get Instance-put Static Static Static Static-get Static-put	Array-put			
Instance-put Static Static-get Static-put Stati			52-5f	
Static Static-get Static-get Static-get Static-put Static-				
Static-get Static-put Sta	•			
Static-put Invoke			60 -72	
Invoke Call a method of kind = {\virtual, super, direct, static, interface} 74 - 78 negative printtype = {\virt, long, byte, float double} Primitive-cast src/destType = {\virt, long, float, double} Tb - 8f No Int-cast destType={\virt, long, float, double} 90 - af Math functions 4 add type={\virt, long, float, double} 90 - af sub type={\virt, long, float, double} 90 - af sub type={\virt, long, float, double} 90 - af div type={\virt, long, float, double} 90 - af				
negative primitype = { int, long, byte, float double } Primitive-cast src/destType = {int, long, float, double} 7b - 8f No Int-cast destType={byte, char, short } 90 - af Math functions 90 - af add type={int, long, float, double} sub type={int, long, float, double} mul type={int, long, float, double} div type={int, long, float, double} rem type={int, long, float, double} and type={int, long, float, double} and type={int, long, float, double} shr ints only shr bo-cf No add type={int, long, float, double}/2 addr sub type={int, long, float, double}/2 addr mul type={int, long, float, double}/2 addr			<u> </u>	
Primitive-cast src/destType = {int, long, float, double} 7b - 8f No Int-cast destType={byte, char, short}			74 – 78	
Int-cast destType={ byte, char, short }			71- 01	Nic
Math functions 90 -af add type={int, long, float, double} sub type={int, long, float, double} mul type={int, long, float, double} div type={int, long, float, double} rem type={int, long, float, double} and type={int, long} or type={int, long} xor ints only shr ints only ushr b0 -cf No of Add type={int, long, float, double/2 addr div type={int, long, float, double/2 addr div type={int, long, float, double/2 addr rem type={int, long, float, double/2 addr xor type={int, long,		, , ,	7b – 8f	No
add type={int, long, float, double} sub type={int, long, float, double} mul type={int, long, float, double} div type={int, long, float, double} rem type={int, long, float, double} and type={int, long, float, double} or type={int, long} or type={int, long} shl ints only shl ints only shr ints only LL.F.D/2 addr bo-cf No add type={int, long, float, double}/2 addr sub type={int, long, float, double}/2 addr mul type={int, long, float, double}/2 addr div type={int, long, float, double}/2 addr rem type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr or type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr sub type={int, long, float, double}/2 addr frem type={int, long, float, double}/2 addr or type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr		dest type={ byte, char, short }	00 (
sub type={int, long, float, double} mul type={int, long, float, double} div type={int, long, float, double} rem type={int, long, float, double} and type={int, long} cor type={int, long} xor ints only shI ints only ushr ints only I,L,F,D/2 addr b0 -cf No add type={int, long, float, double}/2 addr double}/2 addr gub type={int, long, float, double}/2 addr double}/2 addr mul type={int, long, float, double}/2 addr double}/2 addr rem type={int, long, float, double}/2 addr double}/2 addr and type={int, long, float, double}/2 addr double}/2 addr xor type={int, long, float, double}/2 addr double}/2 addr shI type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr		type=fint long float double)	90 -at	
mul type={int, long, float, double} div type={int, long, float, double} rem type={int, long, float, double} and type={int, long} or type={int, long} xor ints only shl ints only ushr ints only ushr ints only l,L,F,D/2 addr b0 -cf add type={int, long, float, double}/2 addr sub type={int, long, float, double}/2 addr mul type={int, long, float, double}/2 addr div type={int, long, float, double}/2 addr rem type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr or type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr				
div type={int, long, float, double} rem type={int, long, float, double} and type={int, long} or type={int, long} xor ints only shI ints only shr ints only ushr ints only I,L,F,D/2 addr b0 -cf add type={int, long, float, double}/2 addr sub type={int, long, float, double}/2 addr mul type={int, long, float, double}/2 addr div type={int, long, float, double}/2 addr rem type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr xor type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr			1	
rem type={int, long, float, double} and type={int, long} or type={int, long} xor ints only shl ints only ushr ints only I,L,F,D/2 addr bype={int, long, float, double}/2 addr sub type={int, long, float, double}/2 addr mul type={int, long, float, double}/2 addr div type={int, long, float, double}/2 addr rem type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr or type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr xor type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr				
and type={int, long} or type={int, long} xor ints only shl ints only shr ints only ushr ints only I,L,F,D/2 addr b0 -cf No add type={int, long, float, double}/2 addr sub type={int, long, float, double}/2 addr mul type={int, long, float, double}/2 addr div type={int, long, float, double}/2 addr rem type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr or type={int, long, float, double}/2 addr xor type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr				
or type={int, long} xor ints only shl ints only shr ints only ushr ints only I,L,F,D/2 addr b0 -cf No add type={int, long, float, double}/2 addr sub type={int, long, float, double}/2 addr mul type={int, long, float, double}/2 addr div type={int, long, float, double}/2 addr rem type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr or type={int, long, float, double}/2 addr xor type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr				
xor ints only			-	
shl ints only shr ints only ushr ints only I,L,F,D/2 addr b0 -cf add type={int, long, float, double}/2 addr sub type={int, long, float, double}/2 addr mul type={int, long, float, double}/2 addr div type={int, long, float, double}/2 addr rem type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr or type={int, long, float, double}/2 addr xor type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr			1	
shr ints only				
ushr ints only I,L,F,D/2 addr b0 -cf No add type={int, long, float, double}/2 addr sub type={int, long, float, double}/2 addr mul type={int, long, float, double}/2 addr div type={int, long, float, double}/2 addr rem type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr or type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr		-	1	
I,L,F,D/2 addr		-		
add type={int, long, float, double}/2 addr sub type={int, long, float, double}/2 addr mul type={int, long, float, double}/2 addr div type={int, long, float, double}/2 addr rem type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr or type={int, long, float, double}/2 addr xor type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr type={int, long, float, double}/2 addr			b0 -cf	No
sub type={int, long, float, double}/2 addr mul type={int, long, float, double}/2 addr div type={int, long, float, double}/2 addr rem type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr or type={int, long, float, double}/2 addr xor type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr type={int, long, float, double}/2 addr		tvpe={int. long. float. double}/2 addr		-
mul type={int, long, float, double}/2 addr div type={int, long, float, double}/2 addr rem type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr or type={int, long, float, double}/2 addr xor type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr				
div type={int, long, float, double}/2 addr rem type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr or type={int, long, float, double}/2 addr xor type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr type={int, long, float, double}/2 addr				
rem type={int, long, float, double}/2 addr and type={int, long, float, double}/2 addr or type={int, long, float, double}/2 addr xor type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr type={int, long, float, double}/2 addr				
and type={int, long, float, double}/2 addr or type={int, long, float, double}/2 addr xor type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr type={int, long, float, double}/2 addr			1	
or type={int, long, float, double}/2 addr xor type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr				
xor type={int, long, float, double}/2 addr shl type={int, long, float, double}/2 addr shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr				
shltype={int, long, float, double}/2 addrshrtype={int, long, float, double}/2 addrushrtype={int, long, float, double}/2 addr				
shr type={int, long, float, double}/2 addr ushr type={int, long, float, double}/2 addr				
ushr type={int, long, float, double}/2 addr				
	Int/16 & int/8	, , , , , , , , , , , , , , , , , , ,	d0 -e2	No

6) Software shall present results of interpreter processing No functional requirements:

None Provided