<?xml version="1.0" encoding="UTF-8"?>

<Program>

<FunctionDeclaration name="\_^main^\_^main" return\_type="INT" last\_temp="108" last\_label="105">

<Blocks>

<Block entry\_label="105"> #block begin with entry label

<Sequence>

<Label label="105"/> #entry label

<Move> #intialize array “a” (size = 4), t100 is “a”

<Temp type="PTR" temp="100"/>

<ExtCall extfun="malloc" type="PTR">

<Arguments>

<Const value="20"/> #((4+1)\*4): (note: the use of int\_length from config.h)

</Arguments>

</ExtCall>

</Move>

<Move> #put size into the first slot of “a”

<Memory>

<Temp type="PTR" temp="100"/>

</Memory>

<Const value="4"/>

</Move>

<Move>

<Temp type="INT" temp="101"/> #stores the length of the input array

<ExtCall extfun="getarray" type="INT"> #get array: external call

<Arguments>

<Temp type="PTR" temp="100"/> #to store the array address (“a”)

</Arguments>

</ExtCall>

</Move>

<Move>

<Temp type="PTR" temp="100"/> # get the result of a = -a

<ESeq>

<Sequence>

<Move>

<Temp type="INT" temp="104"/> #get the size of input “a”

<Memory>

<Temp type="PTR" temp="100"/>

</Memory>

</Move>

<Move>

<Temp type="PTR" temp="105"/> #intialize an array of same size as “a”

<ExtCall extfun="malloc" type="PTR"> #malloc ( (size(a)+1)\*4)

<Arguments>

<BinOp op="\*" type="INT">

<BinOp op="+" type="INT">

<Temp type="INT" temp="104"/> #size of “a”

<Const value="1"/> #+1

</BinOp>

<Const value="4"/> #\*4

</BinOp>

</Arguments>

</ExtCall>

</Move>

<Move> #put the size of “a” in the first slot of the new array

<Memory>

<Temp type="PTR" temp="105"/> #new array

</Memory>

<Temp type="INT" temp="104"/> #size of “a”

</Move>

<Move> #start to do -a. First initialize an index (t106) = 4

<Temp type="INT" temp="106"/> #this is to be used as the index

<Const value="4"/>

</Move>

<Move>

<Temp type="INT" temp="107"/> # bound of the loop = (size+1)\*4

<BinOp op="\*" type="INT">

<BinOp op="+" type="INT">

<Temp type="INT" temp="104"/>

<Const value="1"/>

</BinOp>

<Const value="4"/>

</BinOp>

</Move>

<Label label="100"/>

<CJump relop="&lt;" true="101" false="102"> #test if index<bound

<Temp type="INT" temp="106"/>

<Temp type="INT" temp="107"/>

</CJump>

<Label label="101"/> #still in bound

<Move> #store 0-a[i] into new array[i]

<Memory> #new array at the position index

<BinOp op="+" type="PTR">

<Temp type="PTR" temp="105"/> #new array

<Temp type="INT" temp="106"/> #index

</BinOp>

</Memory>

<BinOp op="-" type="INT"> #do: 0-a[i]

<Const value="0"/>

<Memory>

<BinOp op="+" type="PTR">

<Temp type="PTR" temp="100"/> #”a”

<Temp type="INT" temp="106"/> #index

</BinOp>

</Memory>

</BinOp>

</Move>

<Move> #increment index by 4

<Temp type="INT" temp="106"/>

<BinOp op="+" type="INT">

<Temp type="INT" temp="106"/>

<Const value="4"/>

</BinOp>

</Move>

<Jump label="100"/> #continue the loop

<Label label="102"/> #loop done

</Sequence>

<Temp type="PTR" temp="105"/> #new array (to assign to “a”)

</ESeq>

</Move> #call putarray

<ExpressionStatement>

<ExtCall extfun="putarray" type="INT">

<Arguments>

<Temp type="PTR" temp="100"/> #first argument is the array variable

<Const value="4"/> #second is the number of elements to “put”

</Arguments>

</ExtCall>

</ExpressionStatement>

<Return> #return a[0]. Note we have code to check if 0 is out of bound.

<Memory>

<BinOp op="+" type="PTR">

<Temp type="PTR" temp="100"/>

<BinOp op="\*" type="INT">

<BinOp op="+" type="INT">

<ESeq>

<Sequence>

<Move>

<Temp type="INT" temp="108"/>

<Memory>

<Temp type="PTR" temp="100"/>

</Memory>

</Move>

<CJump relop="&gt;=" true="103" false="104"> #index (0) >= size?

<Const value="0"/>

<Temp type="INT" temp="108"/>

</CJump>

<Label label="103"/>

<ExpressionStatement>

<ExtCall extfun="exit" type="INT">

<Arguments>

<Const value="-1"/>

</Arguments>

</ExtCall>

</ExpressionStatement>

<Label label="104"/>

</Sequence>

<Const value="0"/> #index=0

</ESeq>

<Const value="1"/> #+1

</BinOp>

<Const value="4"/> # \*4

</BinOp>

</BinOp>

</Memory>

</Return>

</Sequence>

</Block>

</Blocks>

</FunctionDeclaration>

</Program>