REPORT COL216 ASSIGNMENT-7

CHIRAG MOHAPATRA
2018CS50403

VISHAL BINDAL
2018CS50425

In this assignment , we read mips instructions from a text file, convert them into 32-character machine instructions and execute them through C++ . We have kept a memory array of 32-char strings (of size 4096) and a register array of integers (of size 32) . Register number 29 is reserved as the stack pointer.

Input/options:

- An input file, currently set to input.txt, for reading the MIPS instructions
- An input file, currently set to delay.txt, for reading the clock cycles used per instruction in the form given in the assignment description
- A variable debug, which if set to true, can be used to execute one instruction at a time and wait for a key press (like enter). If set to false, all instructions are executed non-stop.

Workflow:

- First the input file is scanned and converted to a "clean" file, containing standard delimiter as space, and labels followed by instruction in the same line. This was done so that each label could be associated with a particular instruction no. Thus we can handle cases where the label may or may not be followed by newline(s)
- Then the labels are read from the file and a map is created where each label is mapped to a particular index (denoting index of the instruction following it). A vector of strings is generated where each string denotes the input instructions without the labels.
- On reading an instruction, we parse it first and convert it into a list of tokens . After that, we convert it into binary encoding(R-format/I-format/J-format) accordingly. We store these in the memory array one by one starting from index 0
- The binary instructions are then read from memory and executed accordingly.

The user can also pass register names (like $sp, $ra, $s0, etc) and labels for branching.

Output:

- At each instruction, the PC, instruction in binary, and register file is printed. Whenever a memory location is changed, it is printed.
- At the end, total cycles and IPC are printed.

Testing:

We have done testing on multiple input files , some of which are(Assuming that we initially store 2 in reg 0 and 1 in reg 1 plus reg 29 is 4095):

1	sw $0 , 1024($1) add $0,$0,$0 sub $1 , $0, $1 lw $2, 1022($1)	2
2	sw $0 , 1024($1) sll $0,$0,1 sub $1 , $0, $1 lw $2, 1022($1)	2
3	sw $0 , 1024($1) sll $0,$0,2 srl $0,$0,1 sub $1 , $0, $1 lw $2, 1022($1)	2
4	sw $0 , 1000($0) sw $1 , 1001($0)	5

	add $2, $1, $0 sub $3, $1, $0 sw $2 , 1003($1) sw $3 , 1004($1) lw $4 , 1000($0) lw $5 , 1001($0) lw $6 , 1002($0) lw $7 , 1003($0) add $6, $6, $7 add $5, $5, $6 add $4, $5, $4	
5	sll $0, $0, 1 sll $2, $0, 2 sll $2, $2, 4	256
6	sll $0, $0, 10 srl $2, $0, 5	64
7	sll $0, $0, 3 add $0, $0, $1 srl $0, $0, 1	8
8	sub $0, $1, $0 sll $0, $0, 5	32

9	sll $2, $0, 14 sll $1, $0, 13 sll $0, $0, 5 add $1, $0, $1 add $2, $2, $1	2^15 + 2^14 + 2^6
10	sw $0 , 1024($1) sll $0 ,$0 , 2 sub $2 , $1 . $0 bgtz $2 , first lw $3 , 1017($0) first: blez $2 , second add $3 , $3 , $3 second: j jumping jumping: add $1 , $1 , $1 jal funct add $4 , $0 , $2 j end funct: add $2 , $2 , $2 sll $0 , 1	

	jr $31 end:	
11	sll $4 , $0 , 3 jal acc add $17 , $2 , $16 j end acc: sub $sp , $sp , $0 sw $ra , 1($sp) sw $4 , 0($sp) bgtz $4 , one add $sp , $sp , $0 jr $ra one: sub $4 , $4 , $1 jal acc lw $4 , 0($sp) lw $ra , 1($sp) add $sp , $sp , $0 add $2 , $2 , $4 jr $ra end:	Non leaf procedure Accumulator with n = 16 Assume 1 in $2 and 2 in $0 Argument in $4 $2 is $v0 Sp in $29 $17 contains final answer

12	beq $0 , $1 , one sw $0 , 1024($1) sll $1 , $1 , 1 bne $0 , $1 , one beq $0 , $1 , two one: srl $1 , $1 , 1 two: lw $2 , 1023($1)	
13	sll $4 , $0 , 1 sll $5 , $0 , 2 jal one add $17 , $2 , $16 j end one: sub $29 , $29 , $0 sw $31 , 1($29) sw $4 , 0($29) bgtz $4 , two add $29 , $29 , $0 jr $31 two: sub $4 , $4 , $1 jal one lw $4 , 0($29)	Multiplier using repeated addition Ans is 8*4 = 32 Non leaf procedure

	lw $31 , 1($29) add $29 , $29 , $0 add $2 , $2 , $5 jr $31 end:	