

CPSC 213, Winter 2010, Term 1 — Quiz 3

Date: November 10, 2010; Instructor: Tamara Munzner

NAME: _____ STUDENT NUMBER: _____

1 (22 marks) **Function Pointers / Dynamic Dispatch, Static Control Flow, and Writing Assembly Code.** Consider the following C code that uses function pointers, with an array of pointers and a procedure that uses it to invoke one of them:

```
void procA () { printf ("A"); }
void procB () { printf ("B"); }
void procC () { printf ("C"); }
void (*proc[3]) () = { procA, procB, procC };
int a;
void foo () {
    if (a > 100) proc[0] ();
    else if (a > 0) proc[1] ();
    else if (a == 0) proc[2] ();
}
```

Give SM213 assembly code that implements the procedure call “foo”, assuming that the labels \$a and \$proc have already been created and point to appropriate memory locations for storage. Comment your code.

2 (4 marks) Switch Statements. A `switch` statements can be implemented using a jump table or a sequence of `if` statements.

2a Explain the benefit of using jump tables over `if` statements (just the benefit; not how they work).

2b Are jump tables always the best implementation choice for every `switch` statement? Explain carefully.

3 (6 marks) I/O. Programmed IO (PIO) supports IO-mapped memory, where loads and stores into memory addresses beyond the end of main memory are handled by the I/O controllers. Carefully explain a) what are the limitations of programmed IO and b) how can interrupts be used to solve these problems?

SM213 machine language instructions are 2 bytes or 6 bytes. The first 2 bytes are split into 4 hex digits of 4 bits each for the opcode and the three operands: OpCode, Op0, Op1, Op2. There are 16 possible opcodes, numbered '0' through 'e' in hex. The meaning and use of the three operands is different for each opcode, and is given in the table using these mnemonics: for registers 0-7, 's' for source, 'd' for destination, 'i' for index. The mnemonic 'o' for offset represents an actual number (not a register). Sometimes two hex digits are used to encode this number, sometimes just one. In assembly, 'p' is a multiple of 'o': $o * 2$ or $o * 4$. The placeholder '-' means the hex digit is ignored.

Operation	Machine Language	Semantics/RTL	Assembly
load immediate	0d-- aaaaaaaaa	$r[d] \leftarrow v$	ld aaaaaaaaa, r1
load base+dis	1osd	$r[d] \leftarrow m[o \times 4 + r[s]]$	ld p(rs), rd
load indexed	2sid	$r[d] \leftarrow m[r[i] \times 4 + r[s]]$	ld (rs, ri, 4), rd
store base+dis	3sod	$m[o \times 4 + r[d]] \leftarrow r[s]$	st rs, p(rd)
store indexed	4sdi	$m[r[i] \times 4 + r[d]] \leftarrow r[s]$	st rs, (rd, ri, 4)
halt	f000	(stop execution)	halt
nop	ff00	(do nothing)	nop
rr move	60sd	$r[d] \leftarrow r[s]$	mov rs, rd
add	61sd	$r[d] \leftarrow r[d] + r[s]$	add rs, rd
and	62sd	$r[d] \leftarrow r[d] \& r[s]$	and rs, rd
inc	63-d	$r[d] \leftarrow r[d] + 1$	inc rd
inc addr	64-d	$r[d] \leftarrow r[d] + 4$	inca rd
dec	65-d	$r[d] \leftarrow r[d] - 1$	dec rd
dec addr	66-d	$r[d] \leftarrow r[d] - 4$	deca rd
not	67-d	$r[d] \leftarrow !r[d]$	not rd
shift	7doo	$r[d] \leftarrow r[d] \ll oo$ (if oo is negative)	shl oo, rd shr -oo, rd
branch	8--oo	$pc \leftarrow pc + 2 \times o$	br oo
branch if equal	9doo	if $r[r] == 0$, $pc \leftarrow pc + 2 \times o$	beq rd, oo
branch if greater	adoo	if $r[r] > 0$, $pc \leftarrow pc + 2 \times o$	bgt rd, oo
jump	b--- aaaaaaaaa	$pc \leftarrow a$	jmp aaaaaaaaa
get program counter	6f-d	$r[d] \leftarrow pc$	gpc rd
jump indirect	cdoo	$pc \leftarrow r[r] + 2 \times o$	jmp pp(rd)
jump double ind, b+disp	ddoo	$pc \leftarrow m[4 \times o + r[r]]$	jmp *pp(rd)
jump double ind, index	edi-	$pc \leftarrow m[4 \times r[i] + r[r]]$	jmp *(rd, ri, 4)

Operation	Machine Language Example	Assembly Example
load immediate	0100 00001000	ld \$0x1000, r1
load base+dis	1123	ld 4(r2), r3
load indexed	2123	ld (r1, r2, 4), r3
store base+dis	3123	st r1, 8(r3)
store indexed	4123	st r1, (r2, r3, 4)
halt	f000	halt
nop	ff00	do nothing (nop)
rr move	6012	mov r1, r2
add	6112	add r1, r2
and	6212	and r1, r2
inc	6301	inc r1
inc addr	6401	inca r1
dec	6501	dec r1
dec addr	6601	deca r1
not	6701	not r1
shift	7102	shl \$2, r1
	71fe	shr \$2, r1
branch	1000: 8004	br 0x1008
branch if equal	1000: 9104	beq r1, 0x1008
branch if greater	1000: a104	bgt r1, 0x1008
jump	b000 00001000	jmp 0x1000
get program counter	6f01	gpc r1
jump indirect	c102	jmp 8(r1)
jump double ind, b+disp	d102	jmp *8(r1)
jump double ind, index	e120	jmp *(r1, r2, 4)