CSE30 Final Reference Sheet PID _____

Operator	*Description*	*Associativity*
() [] . ->>> ++ --	Parentheses or function call Brackets or array subscript Dot or Member selection operator Arrow operator Postfix increment/decrement	left to right
++ -- + - ! ~ (type) * & sizeof	Prefix increment/decrement Unary plus and minus not operator and bitwise complement type cast Indirection or dereference operator Address of operator Determine size in bytes	right to left
* / %	Multiplication, division and modulus	left to right
+ -	Addition and subtraction	left to right
<< >>	Bitwise left shift and right shift	left to right
< <= > >=	relational less than/less than equal to relational greater than/greater than or equal to	left to right
== !=	Relational equal to or not equal to	left to right
&&	Bitwise AND	left to right
^	Bitwise exclusive OR	left to right
|	Bitwise inclusive OR	left to right
&&	Logical AND	left to right
||	Logical OR	left to right
?:	Ternary operator	right to left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment operator Addition/subtraction assignment Multiplication/division assignment Modulus and bitwise assignment Bitwise exclusive/inclusive OR assignment	right to left
,	comma operator	left to right

Arithmetic Instructions

add	`add R1, R2, R3`	R1 = R2 + R3
sub	`sub R1, R2, R3`	R1 = R2 - R3
rsb	`rsb R1, R2, R3`	R1 = R3 - R2
mov	`mov R0, R1` `mov R0, imm8`	R0 = R1 R0 = imm8

Push/Pop (list registers from low to high number order)

PUSH	PUSH {R4-R7, FP, LR}	Store the contents of registers R4 through R7, LR and FP on the stack. Update the stack pointer.
POP	POP {R4-R7, FP, LR}	Load from the stack into registers R4 through R7, LR and FP. Update the stack pointer.

Bitwise, Shift and Rotate Instructions

`and`	`and R1, R2, R3`	R1 = R2 & R3 (bitwise AND)
`orr`	`orr R1, R2, R3`	R1 = R2 | R3 (bitwise OR)
`eor`	`eor R1, R2, R3`	R1 = R2 ^ R3 (bitwise Exclusive Or) (EOR)
`bic`	`bic R1, R2, R3`	R1 = R2 & ! R3 (bit clear)
`asr`	`asr R1, R2, const5` `asr r1, r2, r3`	R1 = R2 >> 5 (Arithmetic shift right) (right shift with sign extend) r1 = r2 >> r3
`lsr`	`lsr R1, R2, const5` `lsr r1, r2, r3`	R1 = R2 >> 5 (Logical shift right) (right shift with zero extend) R1 = r2 >> r3
`lsl`	`lsl R1, R2, const5` `lsl r1, r2, r3`	R1 = R2 << 5 (Shift Left) R1 = r2 << r3
`ror`	`ror R1, R2, const`	R1 = R2 bit 0: R2 bit 31-1
`mvn`	`movn R0, R1`	R0 = ~R1 ~ (bitwise NOT) (1's complement)

LDR/STR with Base Register + Immediate Offset Addressing (offset is optional)		
LDR/LDRH/LDRB	LDR R1, [R0] LDR R1, [R0, imm12]	R1 = Mem[R0]; R1 = Mem[R0 +/- imm12];
LDRSH/LDRSB	LDRSH R1, [R0] LDR R1, [R0, imm12]	R1 = Mem[R0]; sign extend value R1 = Mem[R0 +/- imm12]; sign extend value
STR/STRH/STRB	STR R1, [R0] STR R1, [R0, imm12]	Mem[R0] = R1 Mem[R0 +/- imm12] = R1
With Base Register + Register Offset Addressing		
LDR/LDRH/LDRB	LDR R1, [R0, R2]	R1 = Mem[R0 + R2]
LDRSH/LDRSB	LDR R1, [R0, R2]	R1 = Mem[R0 + R2]; sign extend value
STR/STRH/STRB	STR R1, [R0, R2]	Mem[R0 + R2] = R1

Special versions of LDR (using data in .data, .bss or .section .rodata)		
LDR	LDR R0, =LABEL	R0 gets the address of the variable at LABEL
LDR	LDR R0, =0x7f452333	R0 is loaded with the constant 0x7f452333

Control Flow Instructions

`cmp`	`cmp r1, r2`	Set Cond. Flags using R1 - R2
`b`	`b .Label`	Unconditional branch to local .Label
`bl`	`bl Label`	Call to function **normal Label**; save next instruction in lr
`bx`	`bx lr`	Function Call return (PC = LR)
`beq`	`beq .Label`	Branch to local .Label if Equal
`bne`	`bne .Label`	Branch to local .Label if not equal
`ble`	`ble .Label`	Branch to local .Label if Less Than or Equal (signed)
`bls`	`bls .Label`	Branch to local .Label if Less Than or Equal (unsigned)
`blt`	`blt .Label`	Branch to local .Label if Less Than (signed)
`blo`	`blo .Label`	Branch to local .Label if Less Than (unsigned)
`bge`	`bge .Label`	Branch to local .Label if Greater Than or Equal (signed)
`bhs`	`bhs .Label`	Branch to local .Label if Greater Than or Equal (unsigned)
`bgt`	`bgt .Label`	Branch to local .Label if Greater Than (signed)
`bhi`	`bhi .Label`	Branch to local .Label if Greater than (unsigned)
`bmi`	`bmi .Label`	Branch to local .Label if minus/negative
`bpl`	`bpl .Label`	Branch to local .Label if positive or zero (non-negative)
`bvs`	`bvs .Label`	Branch to local .Label if overflow set
`bvc`	`bvc .Label`	Branch to local .Label if overflow is not set
`svc`	`svc 0`	Trap instruction for Linux syscalls; system call number is in R7