Appendix A
LC3

A.1 Installing Pennsim and getting started

To install and use PennSim, read the following documentation :

`http://castle.eiu.edu/~mathcs/mat3670/index/Webview/pennsim-guide.html`

A.2 The LC3 architecture

Memory, Registers The LC-3 memory is shared into words of 16 bits, with address of size 16 bits (from $(0000)_H$ to $(FFFF)_H$).

The LC-3 has 8 main registers : R0, ..., R7. R6 is reserved for the execution stack handling, R7 for the routine return address. They are also specific 16 bits registers: PC (*Program Counter*), IR (*Instruction Register*), PSR (*Program Status Register*).

The PSR has 3 bits N,Z and P that indicate if the last value writen in one of the R0 to R7 registers (viewed as a 16bits 2-complement integer) is strictly negative (N), nul (Z) of strictly positive(P).

Instructions :

Syntax	Action	NZP
NOT DR,SR	DR <- not SR	*
ADD DR,SR1,SR2	DR <- SR1 + SR2	*
ADD DR,SR1,Imm5	DR <- SR1 + SEXT(Imm5)	*
AND DR,SR1,SR2	DR <- SR1 and SR2	*
AND DR,SR1,Imm5	DR <- SR1 and SEXT(Imm5)	*
LEA DR,label	DR <- PC + SEXT(PCoffset9)	*
LD DR,label	DR <- mem[PC + SEXT(PCoffset9)]	*
ST SR,label	mem[PC + SEXT(PCoffset9)] <- SR	
LDR DR,BaseR,Offset6	DR <- mem[BaseR + SEXT(Offset6)]	*
STR SR,BaseR,Offset6	mem[BaseR + SEXT(Offset6)] <- SR	
BR[n][z][p] label	Si (cond) PC <- PC + SEXT(PCoffset9)	
NOP	No Operation	
RET	PC <- R7	
JSR label	R7 <- PC; PC <- PC + SEXT(PCoffset11)	

Assembly directives

`.ORIG add`	Specifies the address where to put the instruction that follows
`.END`	Terminates a block of instructions
`.FILL val`	Reserves a 16-bits word and store the given value at this address
`.BLKW nb`	Reserves nb (consecutive) blocks of 16 bits at this address
`;`	Comments

Predefined interruptions `TRAP` gives a way to implement system calls, each of them is identified by a 8-bit constant. This is handled by the OS of the LC-3. The following macros indicate how to call them:

instruction	macro	description
`TRAP x00`	`HALT`	ends a program (give back decisions to OS)
`TRAP x20`	`GETC`	reads from the keyboard an ASCII char, and puts its value into R0
`TRAP x21`	`OUT`	writes on the screen the ASCII char of R0
`TRAP x22`	`PUTS`	writes on screen the string whose address of first caracter is stored in R0
`TRAP x23`	`IN`	reads from keyboard an ASCII char, outputs on screen and stores its value in R0

Constants : The integer constants encoded in hexadecimal are prefixed by `x`, in decimal by an optional `#` ; they can appear as parameters of the LC-3 instructions (immediate operands, be careful with the sizes) and directives like `.ORIG`, `.FILL` et `.BLKW`.

Coding tricks

- Initialisation to zero of a given register: `AND Ri,Ri,#0`
- Initialisation to a constant `n` (representable on 5 bits in complement to 2):
  `AND Ri,Ri,#0`
  `ADD Ri,Ri,n`
- Computation of the (integer) opposite $R_i \leftarrow (-R_j)$ (1+ complement to 2):
  `NOT Ri,Rj`
  `ADD Ri,Ri,#1`
- Multiplication $R_i \leftarrow 2R_j$: `ADD Ri,Rj,Rj`
- Copy $R_i \leftarrow R_j$: `ADD Ri,Rj,#0`

A.3 LC3 simplified instruction set

Here is a recap of instructions and their encoding:

syntaxe	**action**	**NZP**	**codage**															
			opcode				arguments											
			F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
NOT DR,SR	DR ← not SR	*	1	0	0	1	DR			SR			1	1	1	1	1	1
ADD DR,SR1,SR2	DR ← SR1 + SR2	*	0	0	0	1	DR			SR1			0	0	0	SR2		
ADD DR,SR1,Imm5	DR ← SR1 + SEXT(Imm5)	*	0	0	0	1	DR			SR1			1	Imm5				
AND DR,SR1,SR2	DR ← SR1 and SR2	*	0	1	0	1	DR			SR1			0	0	0	SR2		
AND DR,SR1,Imm5	DR ← SR1 and SEXT(Imm5)	*	0	1	0	1	DR			SR1			1	Imm5				
LEA DR,label	DR ← PC + SEXT(PCoffset9)	*	1	1	1	0	DR			PCoffset9								
LD DR,label	DR ← mem[PC + SEXT(PCoffset9)]	*	0	0	1	0	DR			PCoffset9								
ST SR,label	mem[PC + SEXT(PCoffset9)] ← SR		0	0	1	1	SR			PCoffset9								
LDR DR,BaseR,Offset6	DR ← mem[BaseR + SEXT(Offset6)]	*	0	1	1	0	DR			BaseR			Offset6					
STR SR,BaseR,Offset6	mem[BaseR + SEXT(Offset6)] ← SR		0	1	1	1	SR			BaseR			Offset6					
BR[n][z][p] label	Si (cond) PC ← PC + SEXT(PCoffset9)		0	0	0	0	n	z	p	PCoffset9								
NOP	No Operation		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RET (JMP R7)	PC ← R7		1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0
JSR label	R7 ← PC; PC ← PC + SEXT(PCoffset11)		0	1	0	0	1	PCoffset11										