Appendix – Instruction Set

halt (Stop the CPU and exit):

Machine code: 0x00000000

NOP (No operation):

Machine code: 0x01000000

addi (Add the immediate value and the source register, and store the result to the target register):

Machine code: 0x02xxyyzz

Effect: R[yy] <- R[xx]+zz; PC <- PC+1

Description: R[yy] is R[xx]+zz; PC is incremented by 1

move_reg (Move from the source register to the target register):

Machine code: 0x03xxyy00

Effect: R[yy] <- R[xx]; PC <- PC+1

Description: R[yy] is equal to R[xx]; PC is incremented by 1

movei (Move the immediate value to the target register):

Machine code: 0x0400yyzz

Effect: R[yy] <- zz; PC <- PC+1

Description: R[yy] is equal to zz; PC is incremented by 1

lw (Load a word to the target register from the memory address obtained by the summation of the source register and the immediate value):

Machine code: 0x05xxyyzz

Effect: R[yy] <- M{ R[xx] + zz}; PC <- PC+1

Description: R[yy] is equal to the content of the memory at the address obtained by R[xx]+zz; PC is incremented by 1

sw (Store the content of the target register to the memory address obtained by the summation of the source register and the immediate value):

Machine code: 0x06xxyyzz

Effect: M{R[xx]+zz} <- R[yy]; PC <- PC+1

Description: The content of the memory at the address obtained by R[xx]+zz is equal to R[yy]; PC is incremented by 1

blez (Branch if the source register is less than or equal to zero):

Machine code: 0x07xx00zz

Effect:

```
If R[xx] <= 0
    PC <- PC + 1 + zz
else
    PC <- PC +1
```

Description: If R[xx] is less than or equal to zero, PC is equal to PC+1+zz; otherwise, PC is equal to PC+1.

la: Load address

Assembly: la Ryy, zz
Format: 0x0800yyzz;
Effect: R[yy] <- PC + 1 + zz; PC <- PC+1

add: Add Rxx to Ryy

Assembly: add Rxx, Ryy
Format: 0x0bxxyy00
Effect: R[yy] <- R[xx] + R[yy]; PC <- PC+1

jmp: Unconditional jump

Assembly: jmp zz
Format: 0x0c0000zz
Effect: PC <- PC + 1 + zz

push: Push a register onto the stack

Assembly: push Rxx
Format: 0x09xx0000
Effect: SP <- SP-1;
M{SP} <- R[xx];
PC <- PC+1

pop: Pop a word from the stack to a register

Assembly: pop Ryy
Format: 0x0a00yy00
Effect: R[yy] <- M{SP};
SP <- SP+1; PC <- PC+1

iret: Interrupt return

Assembly: iret
Format: 0x10000000
Effect: PC <- Pop(); PSR <- Pop()

put: Print on screen

Assembly: put Rxx
Format: 0x11xx0000
Effect: print R[xx]; PC<- PC+1