Design Document: Milestone 1 (Revised)
Team 1-6

Our assembly language has
• seven 16-bit general purpose registers (R2-7)
• one special purpose Registers (R0, R1)
R0 – is always set to the value 0
Our language is a load-store architecture, meaning that only load and store instructions access memory. Computation instructions operate only on values in general purpose registers.

We also have
• an INPUTReg
• an OUTPUTReg
• PC or Program Counter
• IPC or Interrupt Program Counter
• Coprocessor Register File

Register Name	Number	Usage
R0	0	Constant 0
R1	1	Temporary (not preserved across call)
R2	2	Temporary (not preserved across call)
R3	3	Temporary (not preserved across call)
R4	4	Temporary (not preserved across call)
R5	5	Temporary (not preserved across call)
R6	6	Temporary (not preserved across call)
R7	7	Return address(used by function)

Assembly Language Instructions

Arithmetic Instructions

Addition
ADD RD, RS
Put the sum of registers RD and RS into register RD.
Shift left logical
LLL RD, C
Takes the data register RD and shifts it left by C.
Subtraction
SUB RD, RS
Put the difference of registers RD and RS into register RD.
Load upper constant
LUC RD, C
Loads the 8-bit constant C in the register RD in the upper 8-bits.

Load lower constant
LLC RD, C
Loads the constant C in to the register RD in the lower 8-bits.

Branch Instructions

Branch if greater than or equal
BT RD, RS, LABEL
Branch to the instruction at LABEL if register RD is greater or equal than register RS.
Branch if equal
EQ RD, RS, LABEL
Branch to the instruction at LABEL if register RD is equal to RS.
Branch to
JP LABEL
Branch to the instruction at LABEL
Branch to and Link
JR LABEL
Branch to the instruction at LABEL and save the next address into R7.
Branch to Register
JRR RD
Branch to the address stored in the register RD.

Load and Store Instructions

Load data
LD RD, RS
Loads the 16-bit word at the address in register RS into register RD.
Store data
ST RD, RS
Store the 16-bit word in register RD to address in register RS

Data Movement Instructions

Move
HMO RD, RS
Copies register RS to register RD.

Interrupts and Input/Output Instructions

Accept Input
AIN RD
Take the data from the INPUTReg to register RD
Assert
AST PORT, RS
Take the data in register RS and output it to the PORT
MaskI
MKI
Toggles IEB
Return from Interrupts
RI

Return to the place in the program where the interrupt occurs.

Instruction Formats

The architecture consists of five different instruction formats.

Arithmetic Instructions

	15 12	11 6	5 3	2 0
ADD	OP	Unused	RD	RS

	15 12	11 3	2 0
ADC	OP	C	RD

	15 12	11 6	5 3	2 0
SUB	OP	Unused	RD	RS

Branch Instructions

	15 12	11 6	5 3	2 0
BT	OP	Label	RD	RS

	15 12	11 6	5 3	2 0
EQ	OP	Label	RD	RS

	15 12	11 0
JP	OP	Label

	15 12	11 3	2 0
JRR	OP	Unused	RD

Load and Store Instructions

	15 12	11 6	5 3	2 0
LD	OP	Unused	RD	RS

	15 12	11 6	5 3	2 0
ST	OP	Unused	RD	RS

Data Movement Instructions

	15 12	11 6	5 3	2 0
HMO	OP	Unused	RD	RS

Interrupts and Input/Output Instructions

	15 12	11 3	2 0
AIN	OP	Target	RS/RD

Instruction	Opcode
ADD	0000
ADC	0001
SUB	0010
BT	0011
EQ	0100
JP	0101
LD	0110
ST	0111
HMO	1000
AIN	1001
AST	1010
MKI	1011
RI	1100
LUC	1101
LLC	1110
JRR	1111

```
INIT:    LUC   R6, INT1(Upper)
         LLC   R6, INT1(Lower)
         AST   PORT1, R6
         LUC   R6, INT2(Upper)
         LLC   R6, INT2(Lower)
         AST   PORT2, R6
         MKI
         HMO  R2, R0
         HMO  R3, R0
         LLC   R3, 2
         HMO R6, R0
INLOOP:  BT R6, R0, INLOOP
         MKI
         JP GCD
INT1:    AIN   R1
         LLC   R2, 8
         ADD  R2, R1
INT2:    LLC R6, 147
GCD:     BT     R0, R3, While
         BT     R2, R3, Else
         HMO R5, R2
```

```
            HMO R2, R3
            HMO R3, R5
            JP  GCD
ELSE:       SUB R2, R3
            JP  GCD
WHILE:      EQ  R2, R1, EXIT
            ADD R3, R1
            JP  GCD
EXIT:       AST PORT3, R3

            # LD  R2, mem [0]
            0110 000000 000 010
            #ADC        R3, 2
            0001 000000000 011
GCD:        #BT  R0, R3, While
            0011 001101 000 010
            #BT  R2, R3, Else
            0011 001011 010 011
            #AIN PORT, R2
            1001 000000000 010
            #LD  R2, mem [0]
            0110 000000 000 010
            #HMO        R5, R2
            1000 000000 101 010
            #HMO        R2, R3
            1000 000000 010 011
            #HMO        R3, R5
            1000 000000 011 101
            #JP   GCD
            0101 000000 000101
ELSE:           #SUB        R2, R3
            0010 000000 010 011
            #JP   GCD
            0101 000000 000101
WHILE:      #EQ  R2, R1, EXIT
            0100 010000 010 001
            #ADD        R3, R1
            0000 000000 011 001
            #JP   GCD
            0101 000000 000101
```

```
EXIT:       #OUT PORT, R3
            1010 000000000 011
```