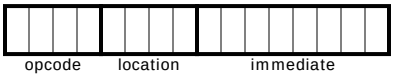
Design Document

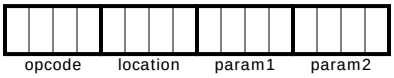
Table of Contents

Addressing (Translating assembly to machine code)

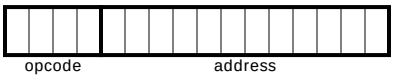
I – Type (immediate)



P – Type (parameter)



J – Type (jump)



Registers

~0 Permanent Zero

This register always contains the logical value zero.

~1 Permanent One

This register always contains the logical value one.

~2 to ~8 Temporary Registers

These registers are available for any part of the program to use for temporary storage.

~9 Input

This is a temporary storage register for processes having to do with the interrupt.

~a IR (Instruction Register)

This is the register for storing instructions.

~b JR (Jump Register)

When performing a jump and link instruction, the location of PC + 2 is stored here.

~c Output

The final result of the computation is stored here, to be output later.

~d PC

This holds the address of the current instruction.

~e Temporary Location

When jumping to a given address, the location to jump to is stored here precedent to the jump command.

~f EPC

Holds the location of an error

Commands

0000 + add (P-type) + ~2, ~3, ~4

Takes the sum of ~3 and ~4 and stores it in ~2.

0001 – subtract (P-type) – ~2, ~3, ~4

Takes the difference of ~3 and ~4 (~3 – ~4) and stores it in ~2.

0010 @ jump and link (J-type) @ 0

Stores PC + 2 to the jump register ~b and jumps to the given immediate address 0.

0011 = branch if equals (P-type) = ~2, ~3, ~4

Compares ~3 and ~4 and if they are equal jumps to the location specified in the temporary location register ~2.

0100 < is less than (P-type) < ~2, ~3, ~4

Compares ~3 and ~4 and if (~3) < (~4) sets the value of ~2 to one.

0101 J jump (J-type) J 0

Jumps to the given immediate address 0.

0110 L load from memory (P-type) L ~2, ~3

Loads the value from memory at location ~3 into the register ~2..

0111 S store to memory (P-type) S ~2, ~3

Stores the value in register ~2 into memory at location ~3.

1000 U load upper (I-type) U ~2, 0

Loads the immediate value 0 into the upper 8 bits of register ~2, ignoring the lower 8 bits.

1001 & AND (P-type) & ~2, ~3, ~4

Stores the bit-wise logical AND of ~3 and ~4 into register ~2.

1010 | OR (P-type) | ~2, ~3, ~4

Stores the bit-wise logical OR of ~3 and ~4 into register ~2.

1011 ! NOT (P-type) ! ~2, ~3

Stores the bit-wise logical NOT of ~3 into register ~2

1100 I input (I-type) I ~2, 0

Loads the immediate value 0 into the lower 8 bits of register ~2, ignoring the upper 8 bits.

1101 R jump register (J-type) R ~e

Jumps to the address in the temporary location register ~e.

1110 M move from IR (R-type) M

Loads the value from external input into the input register ~9.

1111 W write to output (R-type) W

Writes the value currently in the output register ~c to the external output.

Machine Code

```
; The following assembly language and machine code program performs Euclid's
; Algorithm for a given input value from an external device.  The result is sent
; to an external output register to be displayed on a device.
;
; ~0  The value zero
; ~1  The value one
; ~2  a
; ~3  b
; ~4  c
; ~5  temp
; ~6  m
; ~7  lessthantrue
; ~a  receives external input register value
; ~e  templocation
```

address	Commands		Comments	Machine Code
0 000000000000	I	~6, 1	; m = 1	1100011000000001
2 000000000010	M		; ~a = InputRegister	1110000000000000
4 000000000100	+	~4, ~a, ~0	; c = n (User Input)	0000010010100000
6 000000000110	+	~6, ~6, ~1	; m = m + 1;	0000011001100001
8 000000001000	I	~e, 46	; templocation = 46	1100111000101110
10 000000001010	=	~e, ~4, ~1	; if c = 1 goto 46	0011111001000001
12 000000001100	+	~3, ~7, ~0	; b = m;	0000001101110000
14 000000001110	+	~2, ~4, ~0	; a = c;	0000001001000000
16 000000010000	J	18	; goto 18	0101000000010010
18 000000010010	<	~7, ~0, ~3	; lessthantrue = is(0 < b)	0100011100000011
20 000000010100	I	~e, 44	; templocation = 44	1100111000101100
22 000000010110	=	~e, ~7, ~1	; if 0 < b goto 44	0011111001110001
24 000000011000	<	~7, ~2, ~3	; lessthantrue = is(a < b)	0100011100100011
26 000000011010	I	~e, 40	; templocation = 40	1100111000101000
28 000000011100	=	~e, ~7, ~1	; if a < b goto 40	0011111001110001
30 000000011110	J	32	; goto 32	0101000000100000
32 000000100000	+	~5, ~2, ~0	; temp = a	0000010100100000
34 000000100010	+	~2, ~3, ~0	; a = b	0000001000110000
36 000000100100	+	~3, ~5, ~0	; b = temp	0000001101010000
38 000000100110	J	18	; goto 18	0101000000010010

```
40 000000101000   -     ~2, ~2, ~3  ; a = a - b                  0001001000100011
42 000000101010   J     18          ; goto 18                    0101000000010010

44 000000101100   +     ~4, ~2, ~0  ; c = a                      0000010000100000
46 000000101110   J     6           ; goto 6                     0101000000000110

48 000000110000   -     ~6, ~6, ~1  ; m = m - 1;                 0001011001100001
50 000000110010   W     ~6          ; OutputRegister = m         1111011000000000
52 000000110100                     ; program end
```

```
; This Obfusk8r assembly language and machine code program tests the functionality
; of various processor commands.  External input is given an OR command with the
; value 0000000001010101.  If the resultant value equals 0000000011011101, then the
; program exits after sending the value 1 to the OutputRegister; otherwise, the
; value 0 is sent as output.
;
; ~0 The value zero
; ~1 The value one
; ~2 a = 85
; ~3 b = 221
; ~4 OR result
; ~5 Program Result (0 or 1)
; ~a Receives external input register value
; ~e templocation

 0 000000000000   M                   ; ~a = InputRegister          1110000000000000
 2 000000000010   I     ~2, 85       ; a = 85                      1100000001010101
 4 000000000100   I     ~3, 221      ; b = 221                     1100000011011101
 6 000000000110   |     ~4, ~a, ~2   ; Input | a                   1010010010100010
 8 000000001000   I     ~5, 1        ; Program Result = 1          1100010100000001
10 000000001010   I     ~e, 16       ; templocation = 16           1100111000010000
12 000000001100   =     ~e, ~4, ~3   ; if(OR result == b) goto 16  0011111001000011

14 000000001110   I     ~5, 0        ; Program Result = 0          1100010100000000

16 000000010000   W     ~5           ; OutputRegister = ~5         1111010100000000
18 000000010010                      ; program end
```