

Bit16 Manual

Contents

Bit16 CPU Layout	3
Bit16 Specifications	4
Instructions and Cycle Times	4
Registers	4
RAM Module	5
Format Summary	5
Microcode Summary	6
ALU Opcode Summary	6
Load Operation Summary	6
Jump Instructions Summary	7
Macrocode Summary	7
Micro Instructions Formats	8
Format 0: Jump	8
Operation	8
Examples	8
Format 1: Binary ALU Operation	9
Operation	9
Examples	9
Format 1: Unary ALU Operation	9
Operation	9
Examples	9
Format 1: Binary Operation with Immediate	10
Operation	10
Examples	10
Format 2: Byte Operation	11
Operation	11
Examples	11
Format 3: Register Offset	12
Operation	12
Examples	12

Format 3: Frame Pointer Offset	12
Operation	12
Examples	12
Format 4: Load/Store	14
Operation	14
Examples	14
Format 4: Load/Store from Frame Pointer.....	15
Operation	15
Examples	15
Format 5: Load/Store with Register Offset	16
Operation	16
Examples	16
Format 6: Stack Operation (Push/Pop)	17
Operation	17
Examples	17
Format 7: Load Word	18
Operation	18
Examples	18

Bit16 CPU Layout

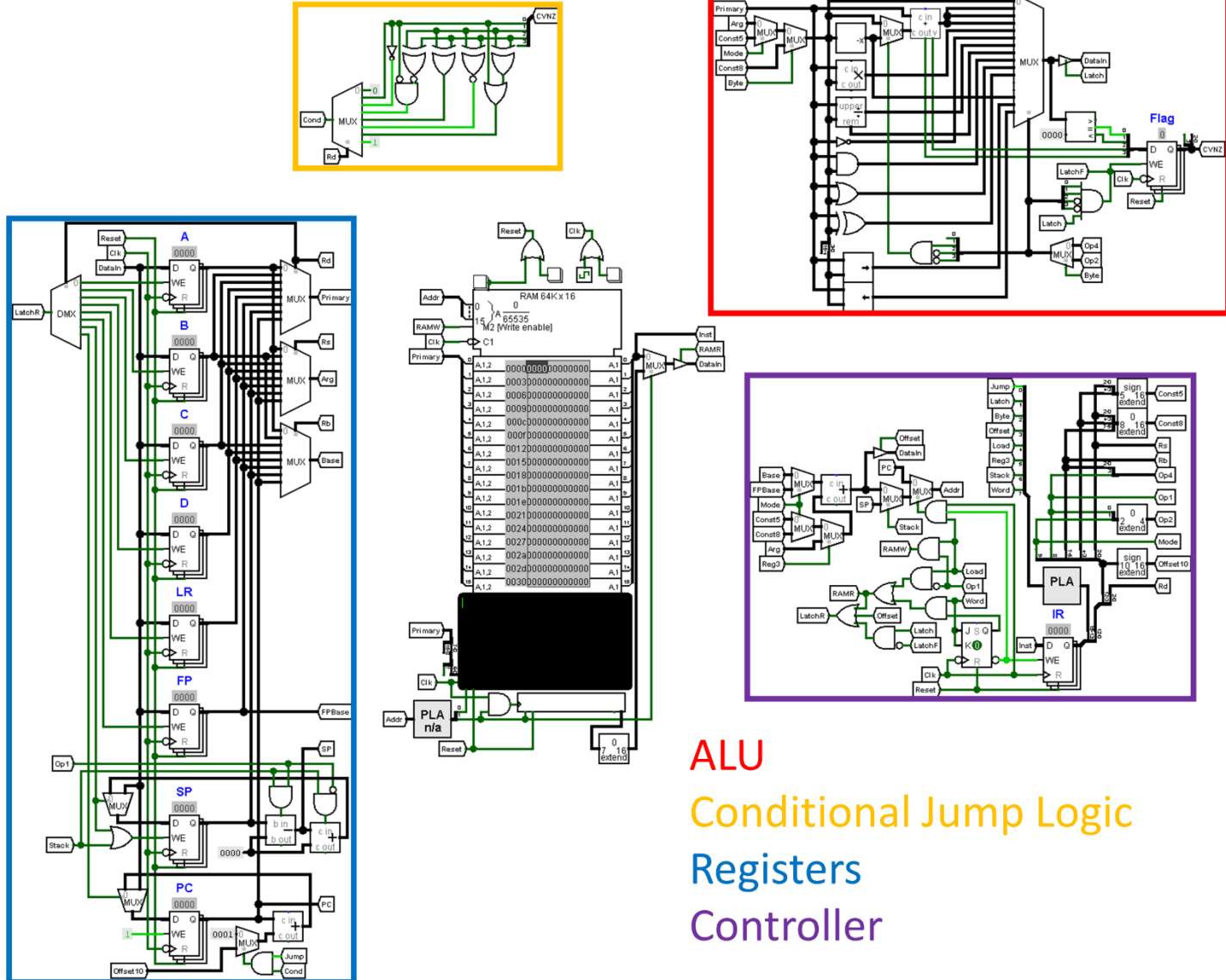


Figure 1: Layout of the bit CPU

Bit16 Specifications

Instructions and Cycle Times

Each instruction is 16 bits long. More information on the formats of each type of 16-bit instruction can be found later in this document. Every microcode instruction has a clock cycle of 1 except for the *load word* instruction which takes 2 clock cycles.

Registers

The bit16 CPU contains addressable registers: 4 general-purpose 16-bit registers and 4 special purpose registers. The general-purpose registers can be used for anything including arithmetic and address calculations. There is also a 16-bit instruction register that stores the current instruction being executed and a 4-bit flag register which stores the different flags from the ALU's operations.

The stack pointer (SP) register is used to keep track of the current top of the stack. The stack starts at the last address in RAM (hex FFFF) and grows downward.

The frame pointer (FP) points to the beginning of the current stack frame and is used to load and store local variables on the stack.

The link register (LR) is used for calling and returning from procedures. LR is 16 bits and should only be used to calculate addresses for the RAM module.

The program counter (PC) register keeps track of the current instruction being executed. The program counter starts at 0 and is incremented after every instruction unless the instruction is a jump instruction in which case, the value in the PC becomes the target for the jump.

Code	Mnemonic	Purpose	Bits	Addressable
0	A	General purpose	16	✓
1	B	General purpose	16	✓
2	C	General purpose	16	✓
3	D	General purpose	16	✓
4	FP	Frame pointer	16	✓
5	SP	Stack pointer	16	✓
6	LR	Link register	16	✓
7	PC	Program counter	16	✓
None	Flag	Flag register	4	
None	IR	Instruction register	16	

Table 1: bit16 CPU registers

RAM Module

The Random-Access-Memory (RAM) module uses full 16-bit addresses. This allows for 65,536 16-bit data/instruction locations. This makes the RAM module roughly 128 kilobytes in size.

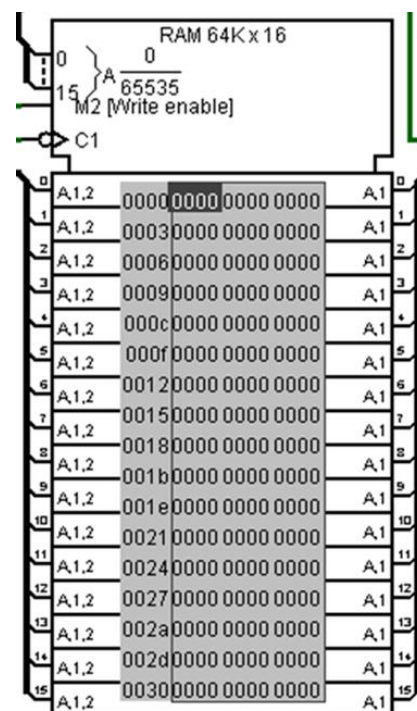


Figure 2: Random-access-memory module

Format Summary

The bit16 instruction set formats are shown in the following figure.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	Offset10										Cond		Jump	
1	0	0	1	0	Op4							Rs		Rd		Binary ALU operation	
1	0	0	1	0	Op1	1	0	1				Rd		Rd		Unary ALU operation	
1	0	0	1	1	Op4				Const5				Rd		Binary ALU operation w/ immediate		
2	0	1	0	Op2	Const8								Rd		Byte operation		
3	0	1	1	0		Rb				Const5				Rd		Register offset	
3	0	1	1	1		Const8								Rd		Frame pointer (FP) byte offset	
4	1	0	0	0	S	Rb				Offset5				Rd		Load/store w/ immediate offset	
4	1	0	0	1	S	Const8								Rd		Load/store from FP w/ byte offset	
5	1	0	1	0	S	Rb							Ro		Rd		Load/store w/ register offset
6	1	1	0		P										Rd		Push/pop register
7	1	1	1												Rd		Load word
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Figure 3: bit16 instruction set formats

Microcode Summary

The following section summarizes the bit16 instruction set microcode.

ALU Opcode Summary

The following table summarizes the bit16 instruction set operations.

Op code	Mnemonic	Instruction	Condition codes set
0	MOV	Move to register	
1	ADD	Add	
2	SUB	Subtract	
3	CMP	Compare	✓
4	MUL	Multiply	
0 (5)	NOT	Bitwise not	
6	DIV	Divide	
7	MOD	Modulo	
8	AND	Bitwise and	
9	OR	Bitwise or	
10	XOR	Bitwise exclusive or	
11			
12			
1 (13)	NEG	Negative	
14	SHR	Shift right	
15	SHL	Shift left	

Table 2: bit16 instruction set opcodes

Load Operation Summary

The following table summarizes the Load operation for the bit16 instruction set.

Mnemonic	Instruction
LD	Load/store

Table 3: bit16 load instruction

Jump Instructions Summary

The following table summarizes the jump codes for the bit16 instruction set.

Code	Mnemonic	Instruction
0	JNV	Jump never
1	JEQ	Jump if equal
2	JNE	Jump if not equal
3	JGT	Jump if greater than
4	JLT	Jump if less than
5	JGE	Jump if greater than or equal to
6	JLE	Jump if less than or equal to
7	JR	Jump relative

Table 4: bit16 jump instructions

Macrocode Summary

The following section summarizes the bit16 instruction set macrocode. Macrocode instructions are instructions that are not supported by the hardware and are made up of one or more microcode instructions.

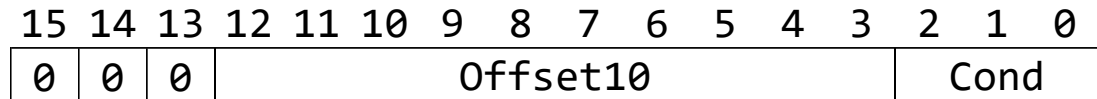
Mnemonic	Instruction	Microcode
PUSH	Push register(s) to stack	PUSH A, B = PUSH A PUSH B
POP	Pop register(s) from stack	POP A, B = POP B POP A
CALL	Call procedure	CALL L = ADD LR, PC, 3 JUMP L
RET	Return from procedure	RET = MOV PC, LR
JUMP	Jump to immediate address	JUMP L = LD PC, L
HALT	Halt execution	HALT = MOV PC, PC

Table 5: bit16 macro code instructions

Micro Instructions Formats

The following section covers the 8 different instruction formats of the bit16 CPU instruction set.

Format 0: Jump



Cond: Condition code

Offset10: Immediate signed 10-bit offset value

Figure 4: Format 0

Operation

Instructions of this format perform a conditional relative jump. This means that the given immediate value is added to the program counter depending on the state of the flag register.

Examples

```
JR    loop    ; Unconditionally jumps to the label "loop"
JNE   .L0     ; Jumps to the label ".L0" if the Z bit is clear
JGE   end     ; Jumps to label "end" if greater than or equal to
```


Format 1: Binary ALU Operation

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	Op4						Rs			Rd		

Rd: Source/destination register

Rs: Source register

Op4: 4-bit opcode

Figure 5: Format 1

Operation

Instructions of this format perform a binary ALU operation on the given registers and place the result in the destination register (Rd) unless it's a comparison operation (CMP, CMN, TST, TEQ).

Examples

ADD A, B ; $A = A + B$

CMP A, B ; Subtract B from A and set the condition codes

OR A, B ; $A = A | B$

TST A, B ; Bitwise AND A and B together and set the condition codes

MOV A, B ; Move the value in B to A

Format 1: Unary ALU Operation

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	Op1	1	0	1			Rd			Rd		

Rd: Source/destination register

Op1: 1-bit opcode. (0 = NOT, 1 = NEG)

Figure 6: Format 1 (Unary)

Operation

Instructions of this format perform a unary ALU operation on the given source register and place the result in the destination register.

Examples

NOT A ; $A = \sim A$

NEG A ; $A = -A$

Format 1: Binary Operation with Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	Op4				Const5					Rd		

Rd: Source/destination register

Const5: Immediate signed 5-bit value

Op4: 4-bit opcode

Figure 7: Format 1 w/ immediate

Operation

Instructions of this format perform a binary ALU operation on the given register and the given immediate value and place the result in the destination register (Rd) unless it's a the comparison operation

Examples

```
ADD    SP, 1      ; SP = SP + 1
CMP     B, 0       ; Subtract 0 from B and set the condition codes
AND     A, 0b111   ; A = A & 0b111
MOV     A, -1      ; Move the value -1 to A
```

Format 2: Byte Operation

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	Op2	Const8								Rd			

Rd: Destination register

Const8: Immediate 8-bit value

Op2: 2-bit opcode (0 = MOV, 1 = ADD,

2 = SUB, 3 = CMP)

Figure 8: Format 2

Operation

Instructions of this format load one 8-bit value into one of the addressable registers.

Examples

```
CMP    A, '\0'    ; Loads the ascii value for 'A' (65) into register A
MOV    B, '\n'    ; Loads the ascii value for '\n' (10) into register B
ADD    B, 'A'     ; Adds the ascii value for 'A' (65) to register B
```

Format 3: Register Offset

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0				Rb								Rd

Rd: Source/destination register

Rb: Base Register

Const5: Immediate signed 5-bit value

Figure 9: Format 3

Operation

Instructions of this format add or subtract a 5-bit signed immediate value to or from the given base register and place the result in the destination register.

Examples

ADD A, B, 0x1 ; A = B + 0x1

SUB A, B, 3 ; A = B - 3

ADD A, A, 1 ; A = A - 1. Equivalent to SUB A, 1

Format 3: Frame Pointer Offset

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1												Rd

Rd: Source/destination register

Const8: Immediate unsigned 8-bit value

Figure 10: Format 3

Operation

Instructions of this format add or subtract an 8-bit unsigned immediate value to or from the frame pointer register and place the result in the destination register.

Examples

ADD A, FP, 1 ; A = FP + 1

ADD B, FP, 0xFF ; B = FP + 0xFF

Format 4: Load/Store

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	S	Rb			Offset5					Rd		

Rd: Source/destination register

Rb: Base register

Offset5: Immediate signed 5-bit offset

S: Store flag (0 = Load, 1 = Store)

Figure 11: Format 4 with immediate offset

Operation

Instructions of this format transfer 16-bit values to and from registers and the RAM module. The load or store operation depends on the S flag. The offset value is added to the value in the base register and sent to the RAM module's address BUS.

Examples

```
LD    [SP, 2], C ; Stores the value in C to the address SP + 2
LD    C, [SP, 2] ; Loads the value at address SP + 2 into C
LD    A, [A]      ; Loads the value at the address in A into A
LD    [B], A      ; Stores the value in A to the address in B
```

Format 4: Load/Store from Frame Pointer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	S	Const8								Rd		

Rd: Source/destination register

Const8: Immediate unsigned 8-bit value

S: Store flag (0 = Load, 1 = Store)

Figure 12: Format 4

Operation

Instructions of this format transfer 16-bit word values to and from registers and the RAM module. The load or store operation depends on the S flag. The offset value is added to the value in the frame pointer register and sent to the RAM module's address BUS.

Examples

```
LD    [FP, 0], A      ; Stores the value in A to the address FP + 0
LD    A, [FP, 4]      ; Loads the value at address FP + 4 into A
LD    A, [FP, 0xFF]   ; Loads the value at address FP + 0xFF into A
```

Format 5: Load/Store with Register Offset

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	S	Rb					Ro			Rd		

Rd: Source/destination register

Rb: Base register

Ro: Offset register

S: Store flag (0 = Load, 1 = Store)

Figure 13: Format 5

Operation

Instructions of this format transfer 16-bit values to and from registers and the RAM module. The load or store operation depends on the S flag. The value in the offset register is added to the value in the base register and sent to the RAM module's address BUS.

Examples

LD [B, C], A ; Stores the value in A to the address B + C

LD A, [B, C] ; Loads the value at address B + C into A

Format 6: Stack Operation (Push/Pop)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0		P											Rd

Rd: Source/destination register

P: Push/pop flag (0 = Pop, 1 = Push)

Figure 14: Format 5

Operation

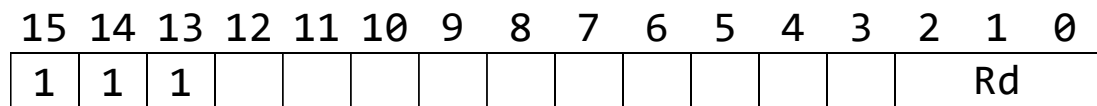
Instructions of this format transfer 16-bit values to and from registers and the RAM module. The load or store operation depends on the S flag. The value in the offset register is added to the value in the base register and sent to the RAM module's address BUS.

Examples

PUSH A ; Pushes A to the stack and decrements SP

POP A ; Pops the top of the stack into A and increments SP

Format 7: Load Word



Rd: Source/destination register

Figure 15: Format 7

Operation

Instructions of this format load a full 16-bit immediate value into one of the addressable registers. The 16-bit value is stored in the next location in memory.

Examples

```
LD    A, 3000    ; Loads the value 3000 from memory into register A
LD    B, =msg     ; Loads the label address into register B
LD    C, 0x7fff   ; Loads the hex value 7fff into register C
```