Bit16 Instruction Set

Contents

Bit16 CPU Layout

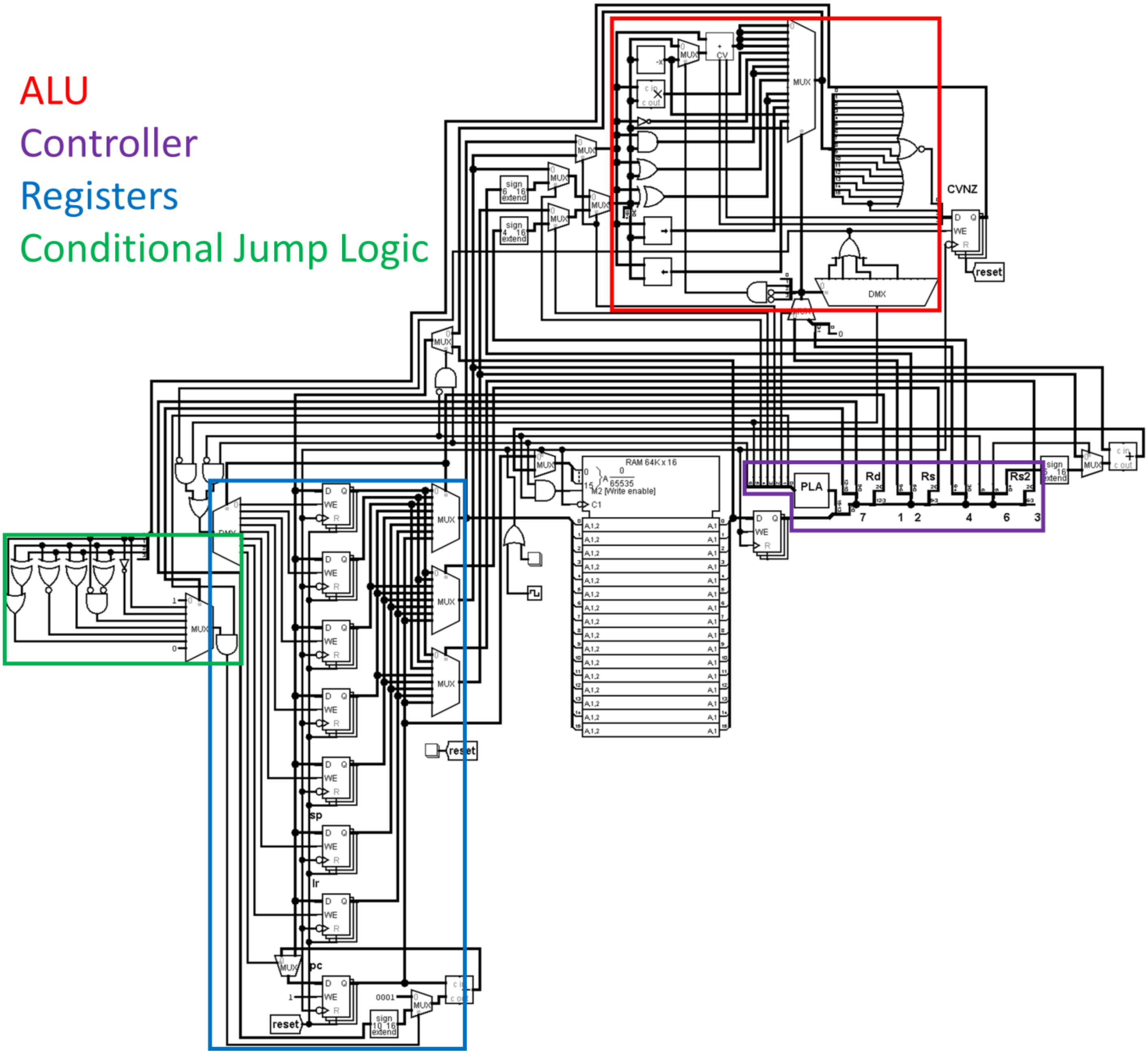

Figure 1: Layout of the bit16 CPU

Bit16 Specifications

Instructions and Their 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 single microcode instruction has a clock cycle of 1.

Registers

The bit16 CPU contains 8 registers: 5 general-purpose 16-bit registers and 3 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 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 ROM 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
0	A	General-purpose	16
1	B		
2	C		
3	D		
4	E		
5	SP	Stack pointer	16
6	LR	Link register	16
7	PC	Program counter	16

Table 1: bit16 CPU registers

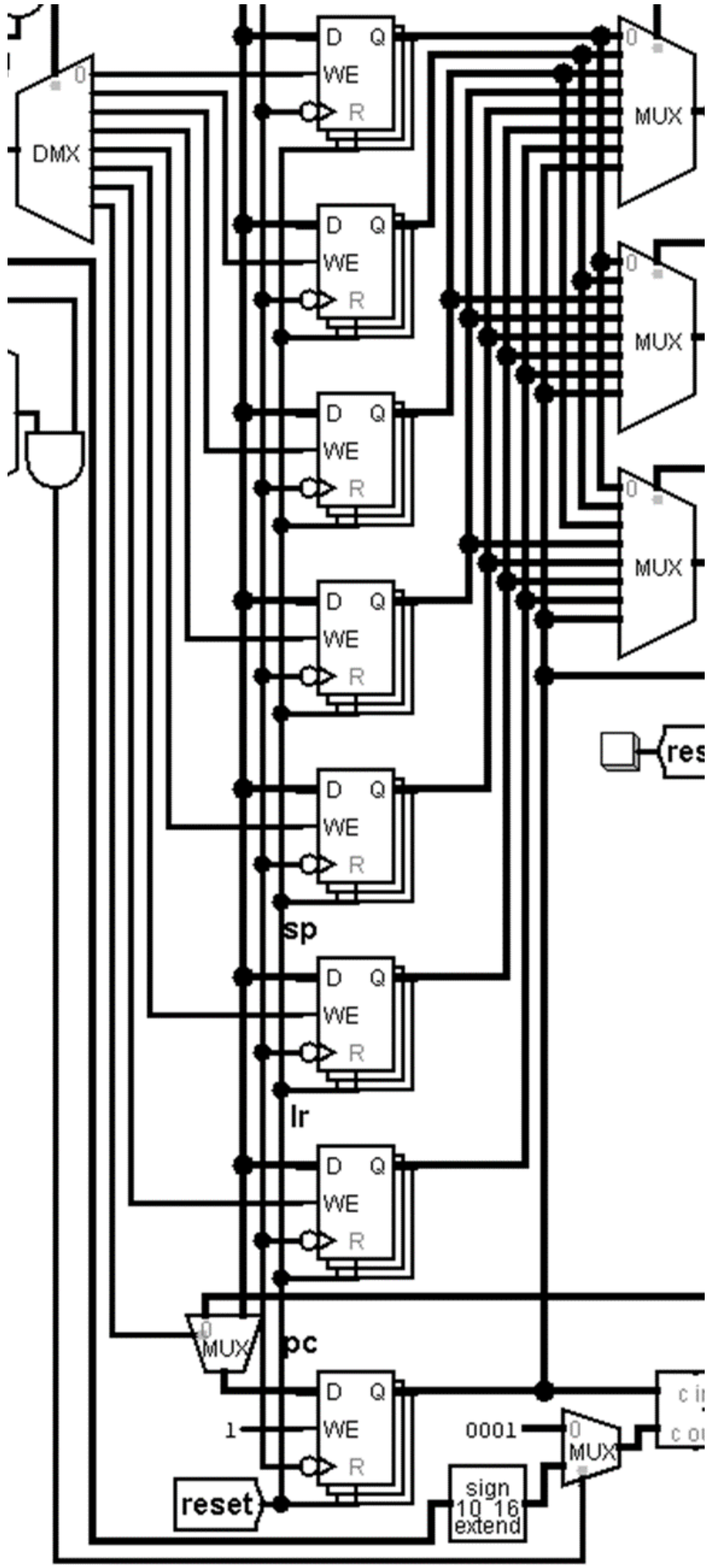

Figure 2: Register module

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.

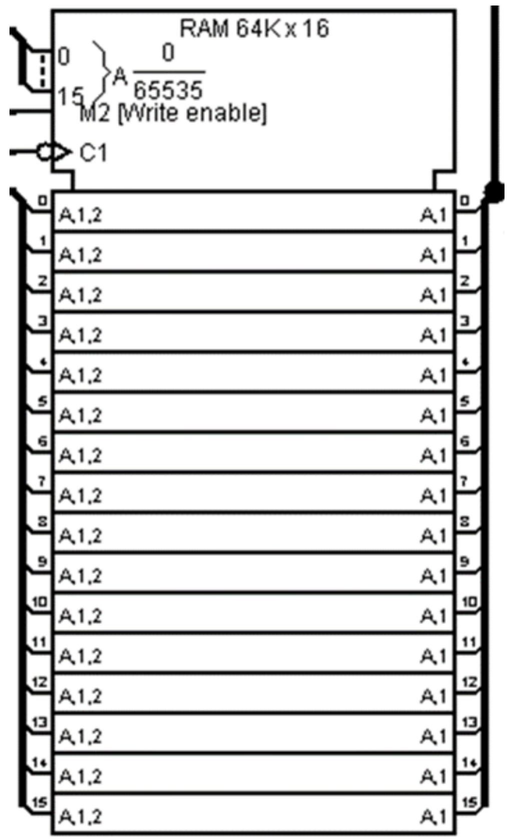

Figure 3: Random-access-memory module

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														No operation
1	0	0	1	Op							Rs			Rd			ALU operation
2	0	1	0	Op				Const6						Rd			ALU w/ immediate operation
3	0	1	1	Op				Rs2			Rs			Rd			ALU operation
4	1	0	0	Op			Const4				Rs			Rd			ALU w/ immediate operation
5	1	0	1	Op										Rd			Unary Operation
6	1	1	0	S	0			Ro			Rb			Rd			Load/store w/ register offset
6	1	1	0	S	1	Offset5					Rb			Rd			Load/store w/ immediate offset
7	1	1	1	Cond			Const10										Jump
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Figure 4: 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.

Op3 code	Op4 code	Mnemonic	Instruction	Condition codes set
0	0	ADD	Add	
	1	CMN	Compare negative	✔
1	2	SUB	subtract	
	3	CMP	Compare	✔
2	4	MUL	Multiply	
	0 (5)	NOT	Bitwise not	
3	6	AND	Bitwise and	
	7	TST	Test bits	✔
4	8	OR	Bitwise or	
5	10	XOR	Bitwise exclusive or	
	11	TEQ	Test bits equality	✔
6	12	SHR	Shift right	
	1 (13)	NEG	Negate	
7	14	SHL	Shift left	
	15	MOV	Move to register	

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.

Jump code	Mnemonic	Instruction
0	JMP	**Jump**
1	JEQ	**J**ump if **eq**ual
2	JNE	**J**ump if **n**ot **e**qual
3	JGT	**J**ump if **g**reater **t**han
4	JLT	**J**ump if **l**ess **t**han
5	JGE	**J**ump if **g**reater than or **e**qual to
6	JLE	**J**ump if **l**ess than or **e**qual to
7	JNV	**J**ump **n**e**v**er

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 = SUB SP, 1` `         LD [SP], A`
POP	Pop register(s) from stack	`POP A = ADD SP, 1` `        LD A, [SP, -1]`

CALL	Call procedure	`CALL L = MOV LR, PC`
		`         ADD LR, 3`
		`         JMP L`

RET	Return from procedure	`RET = MOV PC, LR`
HALT	Halt execution	`HALT = MOV PC, PC`

Table 5: bit16 macrocode instructions

Formats

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

Format 0: No Operation

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

Figure 5: Format 0

Operation

The “nop” instruction does nothing. Nothing is written to a register or a location in memory. The program counter is still incremented.

Format 1: ALU Operation

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

Rd: Source/destination register

Rs: Source register

Op: Opcode

Figure 6: 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 2: ALU Operation with Constant

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	Op				Const6						Rd		

Rd: Source/destination register

Const6: Immediate signed 6-bit value

Op: Opcode

Figure 7: Format2

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 comparison operation (CMP, CMN, TST, TEQ).

Examples

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

Format 3: Binary ALU Operation with Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	Op				Rs2			Rs			Rd		

Rd: Source/destination register

Rs: Source register

Rs2: Source register 2

Op: Opcode

Figure 8: Format 3

Operation

Instructions of this format perform a binary ALU operation on the given source registers (Rs and Rs2) and place the result in the destination register (Rd).

Examples

```
ADD   A, B, C     ; A = B + C
SHL   A, B, C     ; A = B << C
SUB   A, A, B     ; A = A – B. Equivalent to SUB A, B
```

Format 4: Binary ALU Operation with Constant

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	Op			Const4				Rs			Rd		

Rd: Source/destination register

Rs: Source Register

Const4: Immediate signed 4-bit value

Op: Opcode

Figure 9: Format 4

Operation

Instructions of this format perform a binary ALU operation on the given source register and immediate value and place the result in the destination register (Rd).

Examples

```
ADD   A, B, 1     ; A = B + 1
SHR   A, B, 3     ; A = B >> 3
SUB   A, A, xff   ; A = A – 0xff. Equivalent to SUB A, xff
```

Format 5: Unary ALU Operation

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

Rd: Source/destination register

Op: Opcode (0 = NOT, 1 = NEG)

Figure 10: Format 5

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 = ~A
NEG   A     ; A = -A
```

Format 6: Load

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

Rd: Source/destination register

Rb: Base register

Ro: Offset register

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

Figure 11: Format 6

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: Load (with Immediate Offset)

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

Rd: Source/destination register

Rb: Base register

Offset5: Immediate signed 5-bit offset

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

Figure 12: Format 6 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 7: Jump

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	Cond			Const10									

Rd: Source/destination register

Const10: Immediate unsigned 10-bit value

Cond: Condition code

Figure 13: Format 7

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

```
JMP   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
```