

La couche micro-architecture (Chapter 4)

1. Un exemple de micro-architecture

Chemin de données

Micro-instructions

Contrôle de micro-instructions

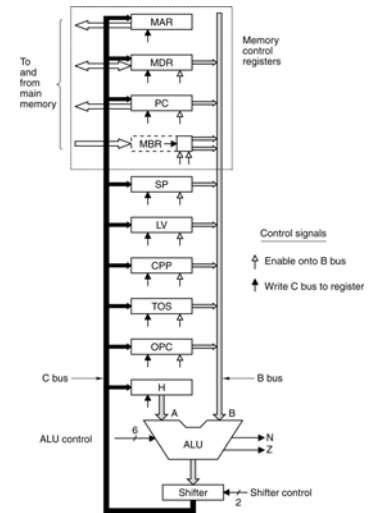
- Exemple de couche ISA: l'IJVM
- Exemple d'implémentation
- Conception d'une micro-architecture
- Amélioration des performances

1

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

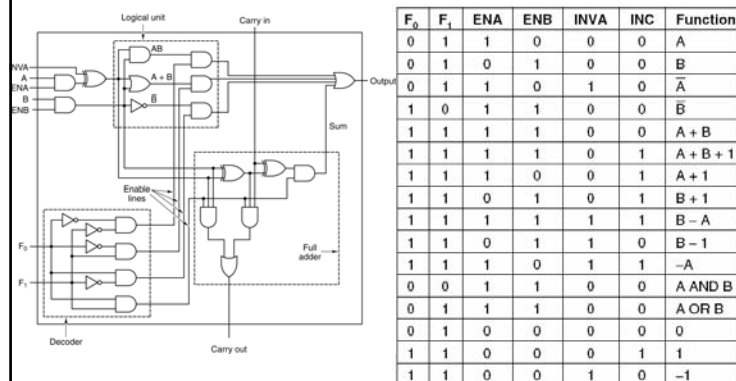
The Data Path (1)

The data path of the example microarchitecture used in this chapter.



Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

The Data Path (2)

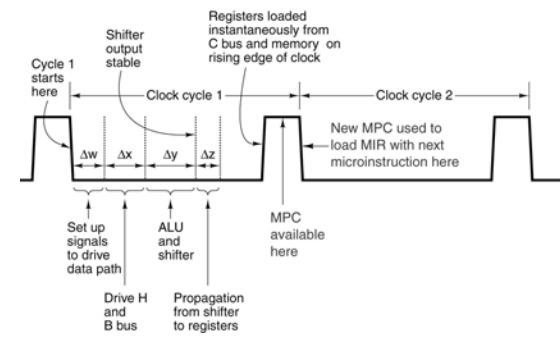


Useful combinations of ALU signals and the function performed.

3

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Data Path Timing

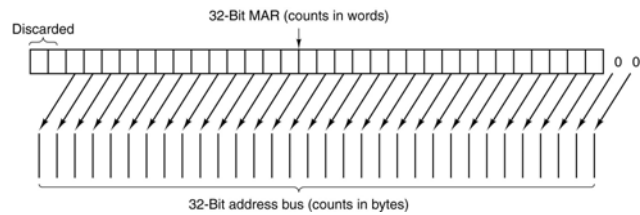


Timing diagram of one data path cycle.

4

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Memory Operation

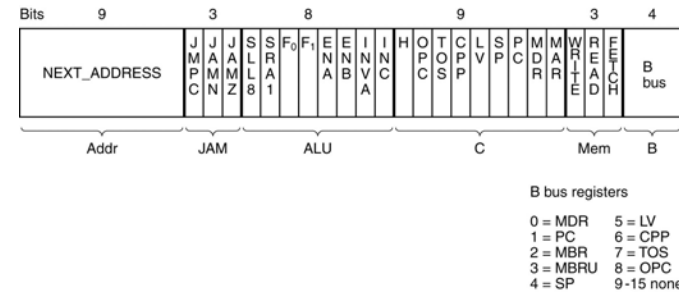


Mapping of the bits in MAR to the address bus.

5

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Microinstructions



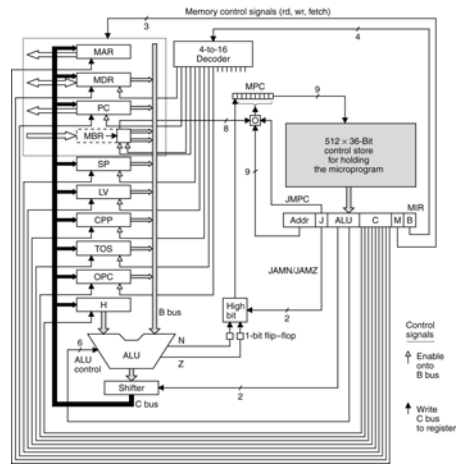
The microinstruction format for the Mic-1.

6

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Microinstruction Control: The Mic-1 (1)

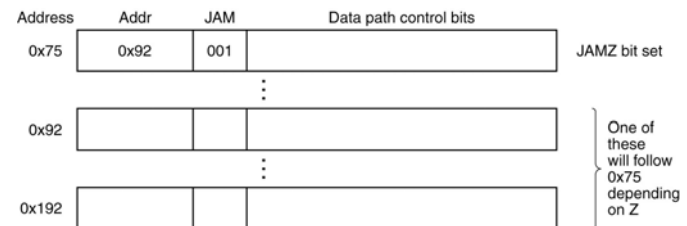
The complete block diagram of our example microarchitecture, the Mic-1.



7

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Microinstruction Control: The Mic-1 (2)



A microinstruction with JAMZ set to 1 has two potential successors.

8

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

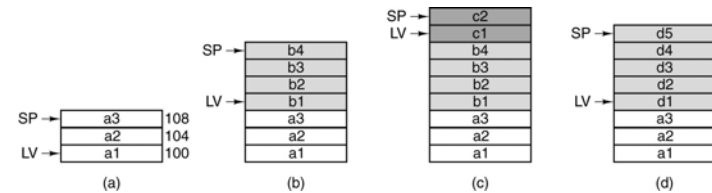
La couche micro-architecture (Chapter 4)

1. Un exemple de Microarchitecture
2. Exemple de couche ISA: l'IJVM
 - Pile
 - Modèle de mémoire
 - Jeu d'instructions
 - Compilation de code Java pour l'IJVM
3. Exemple d'implémentation
4. Conception d'une micro-architecture
5. Amélioration des performances

9

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Stacks (1)



Use of a stack for storing local variables.

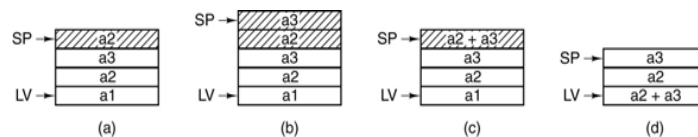
(a) While A is active. (b) After A calls B.

(c) After B calls C. (d) After C and B return and A calls D.

10

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Stacks (2)

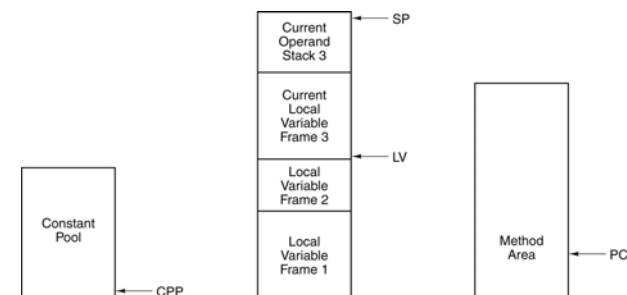


Use of an operand stack for doing an arithmetic computation.

11

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

The IJVM Memory Model



The various parts of the IJVM memory.

12

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

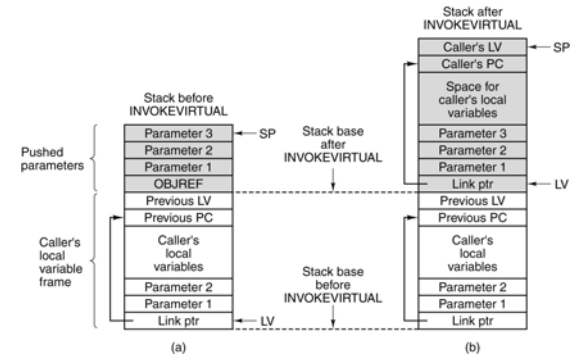
The IJVM Instruction Set (1)

Hex	Mnemonic	Meaning
0x10	BIPUSH <i>byte</i>	Push byte onto stack
0x59	DUP	Copy top word on stack and push onto stack
0xA7	GOTO <i>offset</i>	Unconditional branch
0x60	IADD	Pop two words from stack; push their sum
0x7E	IAND	Pop two words from stack; push Boolean AND
0x99	IFEQ <i>offset</i>	Pop word from stack and branch if it is zero
0x9B	IFLT <i>offset</i>	Pop word from stack and branch if it is less than zero
0x9F	IF_ICMPEQ <i>offset</i>	Pop two words from stack; branch if equal
0x84	IINC <i>varnum const</i>	Add a constant to a local variable
0x15	ILOAD <i>varnum</i>	Push local variable onto stack
0xB6	INVOKEVIRTUAL <i>disp</i>	Invoke a method
0x80	IOR	Pop two words from stack; push Boolean OR
0xAC	IRETURN	Return from method with integer value
0x36	ISTORE <i>varnum</i>	Pop word from stack and store in local variable
0x64	ISUB	Pop two words from stack; push their difference
0x13	LDC <i>W index</i>	Push constant from constant pool onto stack
0x00	NOP	Do nothing
0x57	POP	Delete word on top of stack
0x5F	SWAP	Swap the two top words on the stack
0xC4	WIDE	Prefix instruction; next instruction has a 16-bit index

The IJVM instruction set. The operands *byte*, *const*, and *varnum* are 1 byte. The operands *disp*, *index*, and *offset* are 2 bytes.

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

The IJVM Instruction Set (2)



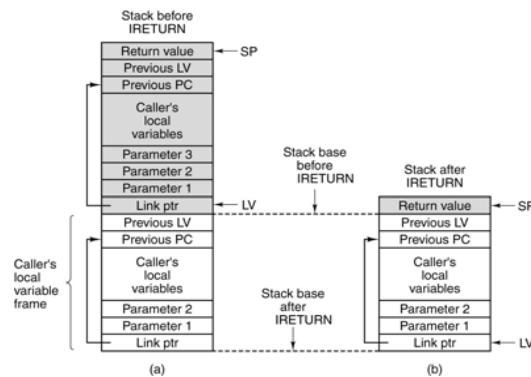
(a) Memory before executing INVOKEVIRTUAL.

(b) After executing it.

14

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

The IJVM Instruction Set (3)



(a) Memory before executing IRETURN.

(b) After executing it.

15

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Compiling Java to IJVM (1)

$i = j + k;$	1	ILOAD j	// $i = j + k$	0x15 0x02
if (i == 3)	2	ILOAD k		0x15 0x03
k = 0;	3	IADD		0x60
else	4	ISTORE i		0x36 0x01
j = j - 1;	5	ILOAD i	// if (i == 3)	0x15 0x01
(a)	6	BIPUSH 3		0x10 0x03
	7	IF_ICMPEQ L1		0x9F 0x00 0x0D
	8	ILOAD j	// $j = j - 1$	0x15 0x02
	9	BIPUSH 1		0x10 0x01
	10	ISUB		0x64
	11	ISTORE j		0x36 0x02
	12	GOTO L2		0xA7 0x00 0x07
	13	L1: BIPUSH 0	// $k = 0$	0x10 0x00
	14	ISTORE k		0x36 0x03
	15	L2:		

(a) A Java fragment.

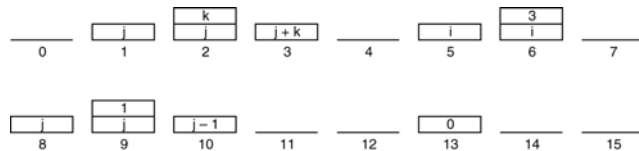
(b) The corresponding Java assembly language.

(c) The IJVM program in hexadecimal.

16

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Compiling Java to IJVM (1)



The stack after each instruction of Fig. 4-14(b).

17

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

La couche micro-architecture (Chapter 4)

1. Un exemple de micro-architecture

2. Exemple de couche ISA: l'IJVM

3. Exemple d'implémentation

Notation des micro-instructions

Implémentation de l'IJVM avec Mic1

4. Conception d'une micro-architecture

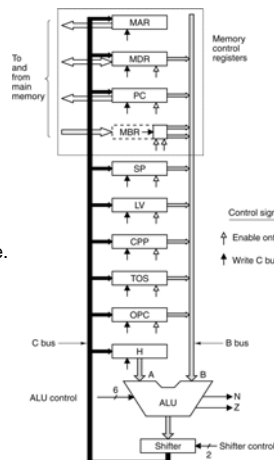
5. Amélioration des performances

18

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Microinstructions and Notation

- Permitted operations
- May be extended by
 - adding "<< 8":
 - shift left by 1 byte.
- A common operation:
 - $H = MBR \ll 8$.



DEST = H
DEST = SOURCE
DEST = H
DEST = SOURCE
DEST = H + SOURCE
DEST = H + SOURCE + 1
DEST = H + 1
DEST = SOURCE + 1
DEST = SOURCE - H
DEST = SOURCE - 1
DEST = -H
DEST = H AND SOURCE
DEST = H OR SOURCE
DEST = 0
DEST = 1
DEST = -1

19

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Implementation of IJVM Using the Mic-1 (1)

Label	Operations	Comments
Main1	PC = PC + 1; fetch; goto (MBR)	MBR holds opcode; get next byte; dispatch
nop1	goto Main1	Do nothing
iadd1	MAR = SP = SP - 1; rd	Read in next-to-top word on stack
iadd2	H = TOS	H = top of stack
iadd3	MDR = TOS = MDR + H; wr; goto Main1	Add two words; write to top of stack
isub1	MAR = SP = SP - 1; rd	Read in next-to-top word on stack
isub2	H = TOS	H = top of stack
isub3	MDR = TOS = MDR - H; wr; goto Main1	Do subtraction; write to top of stack
land1	MAR = SP = SP - 1; rd	Read in next-to-top word on stack
land2	H = TOS	H = top of stack
land3	MDR = TOS = MDR AND H; wr; goto Main1	Do AND; write to new top of stack
lor1	MAR = SP = SP - 1; rd	Read in next-to-top word on stack
lor2	H = TOS	H = top of stack
lor3	MDR = TOS = MDR OR H; wr; goto Main1	Do OR; write to new top of stack
dup1	MAR = SP = SP + 1	Increment SP and copy to MAR
dup2	MDR = TOS; wr; goto Main1	Write new stack word
pop1	MAR = SP = SP - 1; rd	Read in next-to-top word on stack
pop2	TOS = MDR; goto Main1	Wait for new TOS to be read from memory
pop3		Copy new word to TOS
swap1	MAR = SP - 1; rd	Set MAR to SP - 1; read 2nd word from stack
swap2	MAR = SP	Set MAR to top word
swap3	H = MDR; wr	Save TOS in H; write 2nd word to top of stack
swap4	MDR = TOS	Copy old TOS to MDR
swap5	MAR = SP - 1; wr	Set MAR to SP - 1; write as 2nd word on stack
swap6	TOS = H; goto Main1	Update TOS

The microprogram for the Mic-1.

20

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Implementation of IJVM Using the Mic-1 (2)

bipush1	SP = MAR = SP + 1	MBR = the byte to push onto stack
bipush2	PC = PC + 1; fetch	Increment PC, fetch next opcode
bipush3	MDR = TOS = MBR; wr; goto Main1	Sign-extend constant and push on stack
iload1	H = LV	MBR contains index; copy LV to H
iload2	MAR = MBRU + H; rd	MAR = address of local variable to push
iload3	MAR = SP = SP + 1	SP points to new top of stack; prepare write
iload4	PC = PC + 1; fetch; wr	Inc PC; get next opcode; write top of stack
iload5	TOS = MDR; goto Main1	Update TOS
istore1	H = LV	MBR contains index; Copy LV to H
istore2	MAR = MBRU + H	MAR = address of local variable to store into
istore3	MDR = TOS; wr	Copy TOS to MDR; write word
istore4	SP = MAR = SP - 1; rd	Read in next-to-top word on stack
istore5	PC = PC + 1; fetch	Increment PC; fetch next opcode
istore6	TOS = MDR; goto Main1	Update TOS
wide1	PC = PC + 1; fetch;	Fetch operand byte or next opcode
wide2	goto (MBR OR 0x100)	Multway branch with high bit set
wide_load1	PC = PC + 1; fetch	MBR contains 1st index byte; fetch 2nd
wide_load2	H = MBRU << 8	H = 1st index byte shifted left 8 bits
wide_load3	H = MBRU OR H	H = 16-bit index of local variable
wide_load4	MAR = LV + H; rd; goto iload3	MAR = address of local variable to push
wide_store1	PC = PC + 1; fetch	MBR contains 1st index byte; fetch 2nd
wide_store2	H = MBRU << 8	H = 1st index byte shifted left 8 bits
wide_store3	H = MBRU OR H	H = 16-bit index of local variable
wide_store4	MAR = LV + H; goto istore3	MAR = address of local variable to store into
ldc_w1	PC = PC + 1; fetch	MBR contains 1st index byte; fetch 2nd
ldc_w2	H = MBRU << 8	H = 1st index byte << 8
ldc_w3	H = MBRU OR H	H = 16-bit index into constant pool
ldc_w4	MAR = H + CPP; rd; goto iload3	MAR = address of constant in pool

The microprogram for the Mic-1.

21

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Implementation of IJVM Using the Mic-1 (3)

Label	Operations	Comments
ilinc1	H = LV	MBR contains index; Copy LV to H
iinc2	MAR = MBRU + H; rd	Copy LV + index to MAR; Read variable
iinc3	PC = PC + 1; fetch	Fetch constant
iinc4	H = MDR	Copy variable to H
iinc5	PC = PC + 1; fetch	Fetch next opcode
iinc6	MDR = MBR + H; wr; goto Main1	Put sum in MDR; update variable
goto1	OPC = PC - 1	Save address of opcode.
goto2	PC = PC + 1; fetch	MBR = 1st byte of offset; fetch 2nd byte
goto3	H = MBR << 8	Shift and save signed first byte in H
goto4	H = MBRU OR H	H = 16-bit branch offset
goto5	PC = OPC + H; fetch	Add offset to OPC
goto6	goto Main1	Wait for fetch of next opcode
ift1	MAR = SP = SP - 1; rd	Read in next-to-top word on stack
ift2	OPC = TOS	Save TOS in OPC temporarily
ift3	TOS = MDR	Put new top of stack in TOS
ift4	N = OPC; if (N) goto T; else goto F	Branch on N bit
ifeq1	MAR = SP = SP - 1; rd	Read in next-to-top word of stack
ifeq2	OPC = TOS	Save TOS in OPC temporarily
ifeq3	TOS = MDR	Put new top of stack in TOS
ifeq4	Z = OPC; if (Z) goto T; else goto F	Branch on Z bit

The microprogram for the Mic-1.

22

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Implementation of IJVM Using the Mic-1 (4)

if_icmpeq1	MAR = SP = SP - 1; rd	Read in next-to-top word of stack
if_icmpeq2	MAR = SP = SP - 1	Set MAR to read in new top-of-stack
if_icmpeq3	H = MDR; rd	Copy second stack word to H
if_icmpeq4	OPC = TOS	Save TOS in OPC temporarily
if_icmpeq5	TOS = MDR	Put new top of stack in TOS
if_icmpeq6	Z = OPC - H; if (Z) goto T; else goto F	If top 2 words are equal, goto T, else goto F
T	OPC = PC - 1; goto goto2	Same as goto1; needed for target address
F	PC = PC + 1	Skip first offset byte
F2	PC = PC + 1; fetch	PC now points to next opcode
F3	goto Main1	Wait for fetch of opcode
invokevirtual1	PC = PC + 1; fetch	MBR = index byte 1; inc. PC, get 2nd byte
invokevirtual2	H = MBRU << 8	Shift and save first byte in H
invokevirtual3	H = MBRU OR H	H = offset of method pointer from CPP
invokevirtual4	MAR = CPP + H; rd	Get pointer to method from CPP area
invokevirtual5	OPC = PC + 1	Save Return PC in OPC temporarily
invokevirtual6	PC = MDR; fetch	PC points to new method; get param count
invokevirtual7	PC = PC + 1; fetch	Fetch 2nd byte of parameter count
invokevirtual8	H = MBRU << 8	Shift and save first byte in H
invokevirtual9	H = MBRU OR H	H = number of parameters
invokevirtual10	PC = PC + 1; fetch	Fetch first byte of # locals
invokevirtual11	TOS = SP - H	TOS = address of OBJREF - 1
invokevirtual12	TOS = MAR = TOS + 1	TOS = address of OBJREF (new LV)
invokevirtual13	PC = PC + 1; fetch	Fetch second byte of # locals
invokevirtual14	H = MBRU << 8	Shift and save first byte in H
invokevirtual15	H = MBRU OR H	H = # locals

The microprogram for the Mic-1.

23

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Implementation of IJVM Using the Mic-1 (5)

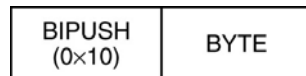
Label	Operations	Comments
invokevirtual16	MDR = SP + H + 1; wr	Overwrite OBJREF with link pointer
invokevirtual17	MAR = SP = MDR;	Set SP, MAR to location to hold old PC
invokevirtual18	MDR = OPC; wr	Save old PC above the local variables
invokevirtual19	MAR = SP = SP + 1	SP points to location to hold old LV
invokevirtual20	MDR = LV; wr	Save old LV above saved PC
invokevirtual21	PC = PC + 1; fetch	Fetch first opcode of new method.
invokevirtual22	LV = TOS; goto Main1	Set LV to point to LV Frame
ireturn1	MAR = SP = LV; rd	Reset SP, MAR to get link pointer
ireturn2		Wait for read
ireturn3	LV = MAR = MDR; rd	Set LV to link ptr; get old PC
ireturn4	MAR = LV + 1	Set MAR to read old LV
ireturn5	PC = MDR; rd; fetch	Restore PC; fetch next opcode
ireturn6	MAR = SP	Set MAR to write TOS
ireturn7	LV = MDR	Restore LV
ireturn8	MDR = TOS; wr; goto Main1	Save return value on original top of stack

The microprogram for the Mic-1.

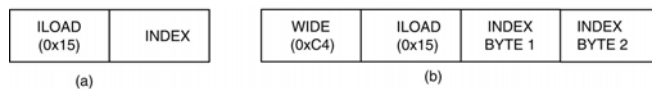
24

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Implementation of IJVM Using the Mic-1 (6)



The BIPUSH instruction format.



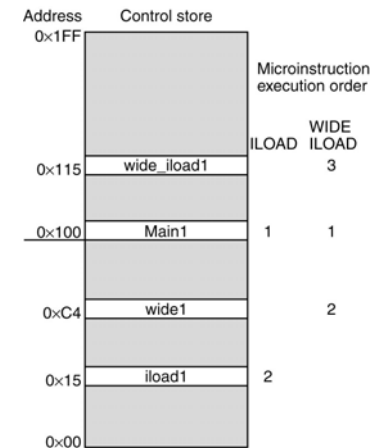
- (a) ILOAD with a 1-byte index.
 (b) WIDE ILOAD with a 2-byte index.

25

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Implementation of IJVM Using the Mic-1 (7)

The initial microinstruction sequence for ILOAD and WIDE ILOAD. The addresses are examples.



Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Implementation of IJVM Using the Mic-1 (8)

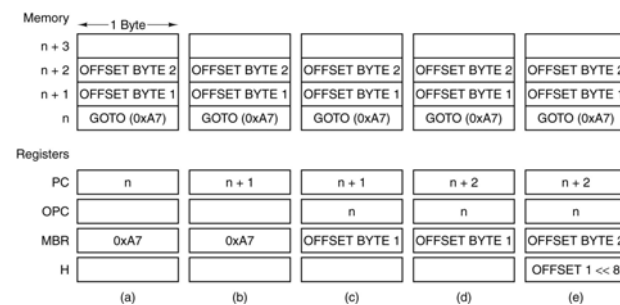


The IINC instruction has two different operand fields.

27

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Implementation of IJVM Using the Mic-1 (9)



The situation at the start of various microinstructions.
 (a) Main1. (b) goto1. (c) goto2. (d) goto3. (e) goto4.

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Merging the Interpreter Loop with the Microcode (1)

Label	Operations	Comments
pop1	MAR = SP = SP - 1; rd	Read in next-to-top word on stack
pop2		Wait for new TOS to be read from memory
pop3	TOS = MDR; goto Main1	Copy new word to TOS
Main1	PC = PC + 1; fetch; goto (MBR)	MBR holds opcode; get next byte; dispatch

Original microprogram sequence for executing POP.

29

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Merging the Interpreter Loop with the Microcode (2)

Label	Operations	Comments
pop1	MAR = SP = SP - 1; rd	Read in next-to-top word on stack
Main1.pop	PC = PC + 1; fetch	MBR holds opcode; fetch next byte
pop3	TOS = MDR; goto (MBR)	Copy new word to TOS; dispatch on opcode

Enhanced microprogram sequence for executing POP.

30

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

A Three Bus Architecture (1)

Label	Operations	Comments
iload1	H = LV	MBR contains index; Copy LV to H
iload2	MAR = MBRU + H; rd	MAR = address of local variable to push
iload3	MAR = SP = SP + 1	SP points to new top of stack; prepare write
iload4	PC = PC + 1; fetch; wr	Inc PC; get next opcode; write top of stack
iload5	TOS = MDR; goto Main1	Update TOS
Main1	PC = PC + 1; fetch; goto (MBR)	MBR holds opcode; get next byte; dispatch

Mic-1 code for executing ILOAD.

31

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

A Three Bus Architecture (2)

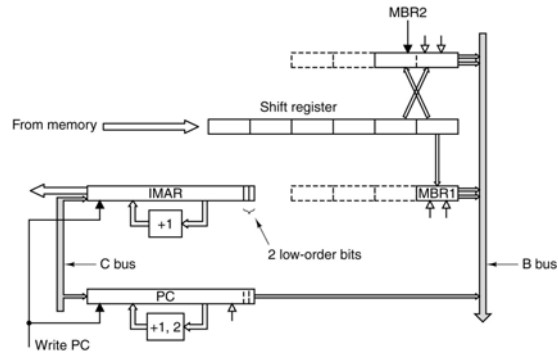
Label	Operations	Comments
iload1	MAR = MBRU + LV; rd	MAR = address of local variable to push
iload2	MAR = SP = SP + 1	SP points to new top of stack; prepare write
iload3	PC = PC + 1; fetch; wr	Inc PC; get next opcode; write top of stack
iload4	TOS = MDR	Update TOS
iload5	PC = PC + 1; fetch; goto (MBR)	MBR already holds opcode; fetch index byte

Three-bus code for executing ILOAD.

32

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

A Three Bus Architecture (3)

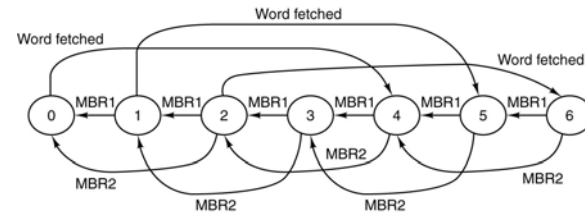


A fetch unit for the Mic-1.

33

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

A Three Bus Architecture (4)



Transitions

MBR1: Occurs when MBR1 is read

MBR2: Occurs when MBR2 is read

Word fetched: Occurs when a memory word is read and 4 bytes are put into the shift register

A finite state machine for implementing the IFU.

34

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Out-of-Order Execution and Register Renaming (1)

Cy	#	Decoded	Iss	Ret	Registers being read								Registers being written							
					0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
1	1	R3=R0+R1	1		1	1									1					
	2	R4=R0+R2	2		2	1	1								1	1				
2	3	R5=R0+R1	3		3	2	1								1	1	1			
	4	R6=R1+R4	—		3	2	1								1	1	1			
3					3	2	1								1	1	1			
4				1	2	1	1													
				2	1	1											1	1		
				3																
5																				
	5	R7=R1+R2	4	5				1											1	1
6	6	R1=R0-R2	—					2	1	1									1	1
7								1	1											1
8				4																
				5																

A superscalar CPU with in-order issue and in-order completion.

35

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Out-of-Order Execution and Register Renaming (2)

Cy	#	Decoded	Iss	Ret	Registers being read								Registers being written							
					0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
9				6			1	1												
	7	R3=R3+R1	—				1	1												
10							1	1												
11																				
				6																
12																				
	8	R1=R4+R4	—				1	1	1											
13							1	1												
14							1	1												
15																				
				7																
16																				
				8																
17									2											
									2											
18																				

A superscalar CPU with in-order issue and in-order completion.

36

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Out-of-Order Execution and Register Renaming (3)

Cy	#	Decoded	Iss	Ret	Registers being read								Registers being written							
					0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
1	1	R3=R0+R1	1		1	1								1						
2	2	R4=R0+R2	2		2	1	1							1	1					
3	3	R5=R0+R1	3		3	2	1						1	1	1					
4	4	R6=R1+R4	—		3	2	1						1	1	1					
5	5	R7=R1+R2	5		3	3	2						1	1	1	1				
6	6	S1=R0-R2	6		4	3	3						1	1	1	1				
				2	3	3	2						1	1	1	1				
4	7	R3=R3+S1	4		3	4	2	1					1	1	1	1				
8	8	S2=R4+R4	8		3	4	2	3					1	1	1	1				
				1	2	3	2	3					1	1	1	1				
				3	1	2	2	3					1	1	1	1				
5				6	2	1		3					1				1	1		
6				7	2	1	1	3					1	1			1	1		
				4	1	1	1	2					1	1			1	1		
				5				1					1							
				8				1					1							
7								1					1							
8								1					1							
9				7									1							

Operation of a superscalar CPU with out-of-order issue and out of-order completion.

37

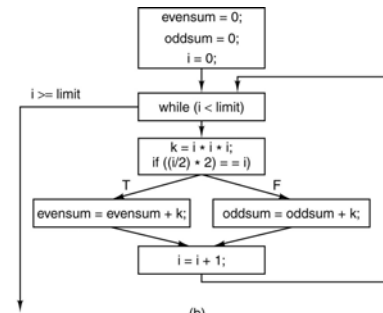
Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Speculative Execution

```

evensum = 0;
oddsum = 0;
i = 0;
while (i < limit) {
    k = i * i * i;
    if (((i/2) * 2) == i)
        evensum = evensum + k;
    else
        oddsum = oddsum + k;
    i = i + 1;
}
    
```

(a)



(b)

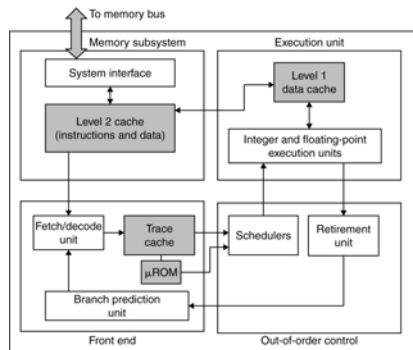
(a) A program fragment.

(b) The corresponding basic block graph.

38

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Overview of the NetBurst Microarchitecture



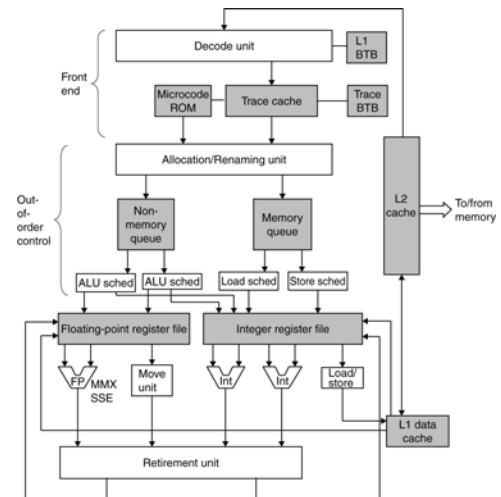
The block diagram of the Pentium 4.

39

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

The NetBurst Pipeline

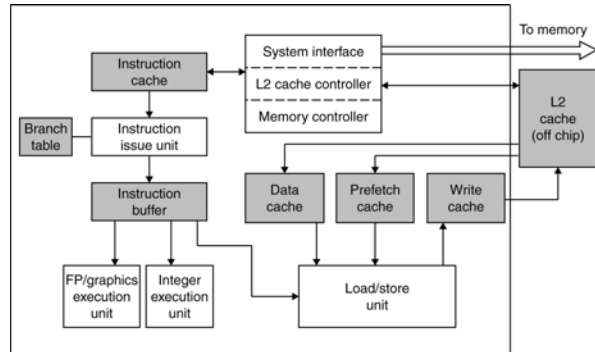
A simplified view of the Pentium 4 data path.



40

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

Overview of the UltraSPARC III Cu Microarchitecture



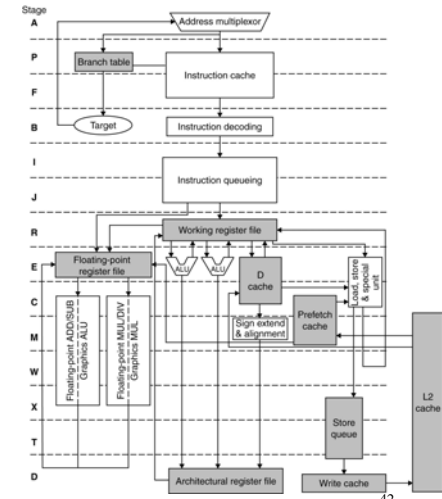
The block diagram of the UltraSPARC III Cu.

41

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

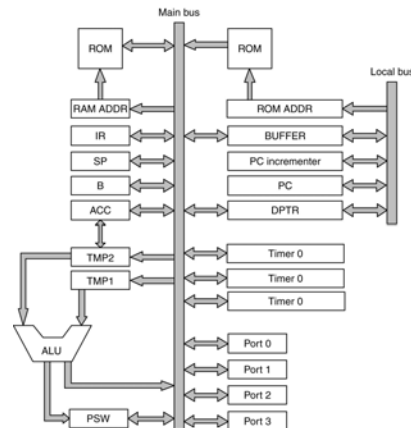
UltraSPARC III Cu Pipeline

A simplified representation of the UltraSPARC III Cu pipeline.



Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0

The Microarchitecture of the 8051 CPU



The microarchitecture of the 8051.

43

Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc. All rights reserved. 0-13-148521-0