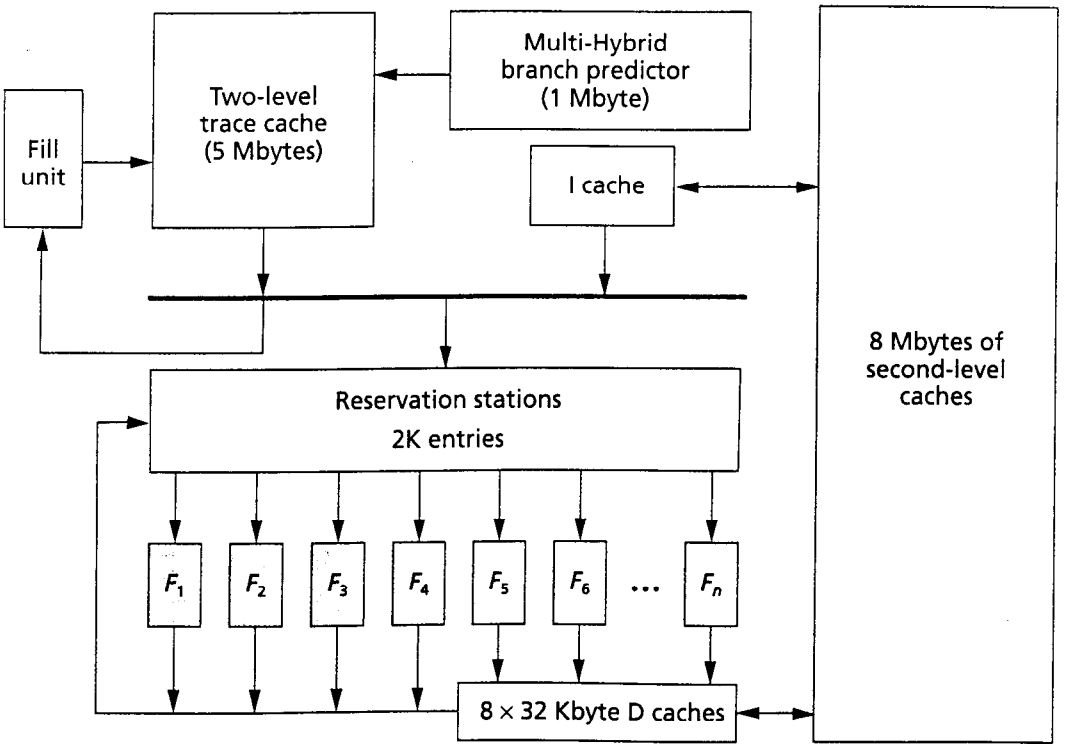
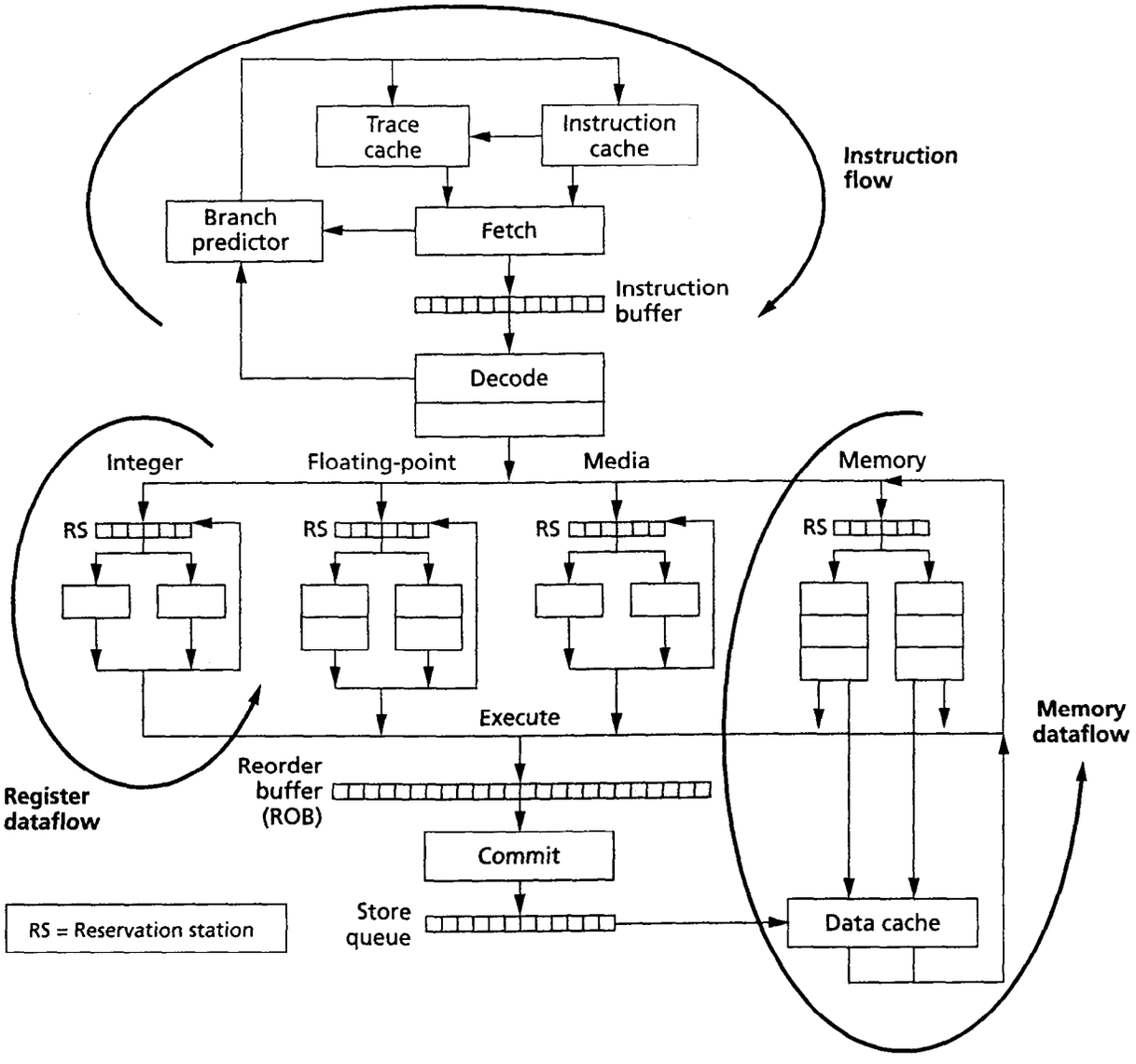
1 Uniprocessor Fig. 1, p. 52



A SSS

2 Superspeculative - Superflow Fig. 2, p. 63



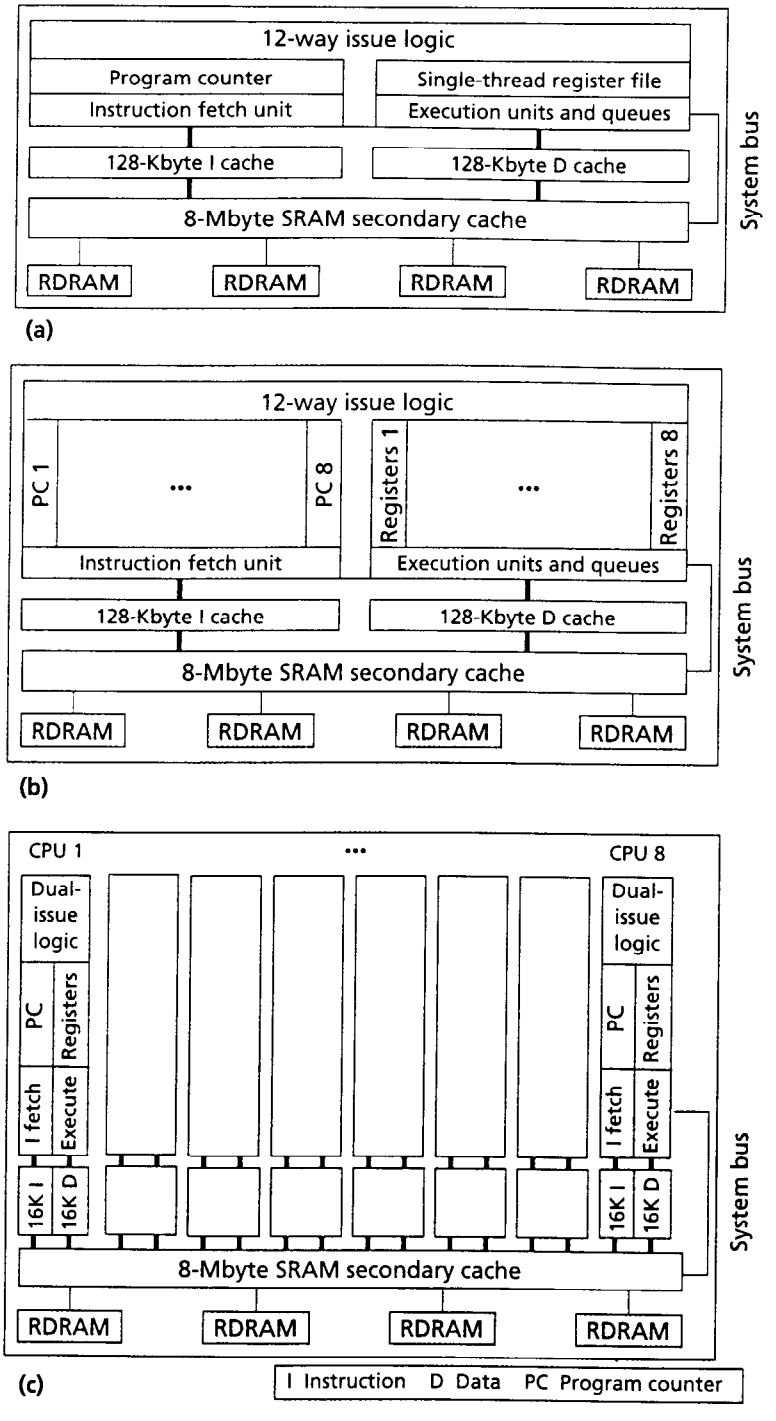
A. SSS

Table 1. Estimated transistor cost of Superflow.

Resource	Description	Transistor count (millions)
CPU core logic	(32 instructions issued per cycle / 4 instructions issued per cycle)^2 × 2 million transistors in a PowerPC 604, which issues four instructions per cycle	128.0
Value prediction table	32 Kbytes × 8 bits/byte × 6 transistors/SRAM bit	1.6
Classification table	8K entries × 2 bits/entry × 6 transistors/SRAM bit	0.1
Dependence prediction table	8K entries × 7 bits/entry × 64 ports × 6 transistors/SRAM bit	22.0
Alias prediction table	8K entries × 7 bits/entry × 32 ports × 6 transistors/SRAM bit	11.0
Pattern history tables	64K entries × 2 bits/entry × 2 tables × 6 transistors/SRAM bit	1.6
Trace cache	64 Kbytes (estimated size) × 8 bits/byte × 6 transistors/SRAM bit	3.1
Level 1 instruction cache	64 Kbytes × 8 bits/byte x 6 transistors/SRAM bit	3.1
Level 1 data cache	64 Kbytes × 4 ports × 8 bits/byte × 6 transistors/SRAM bit	12.6
Processor core total		**183.1**
Level 2 unified cache	16 Mbytes × 8 bits/byte × 6 transistors/SRAM bit (approximately)	805.3
Grand total		**988.4**

T. 1, p. 69

6. CMP Fig. 1, p 81



A. SSS

B. SMT

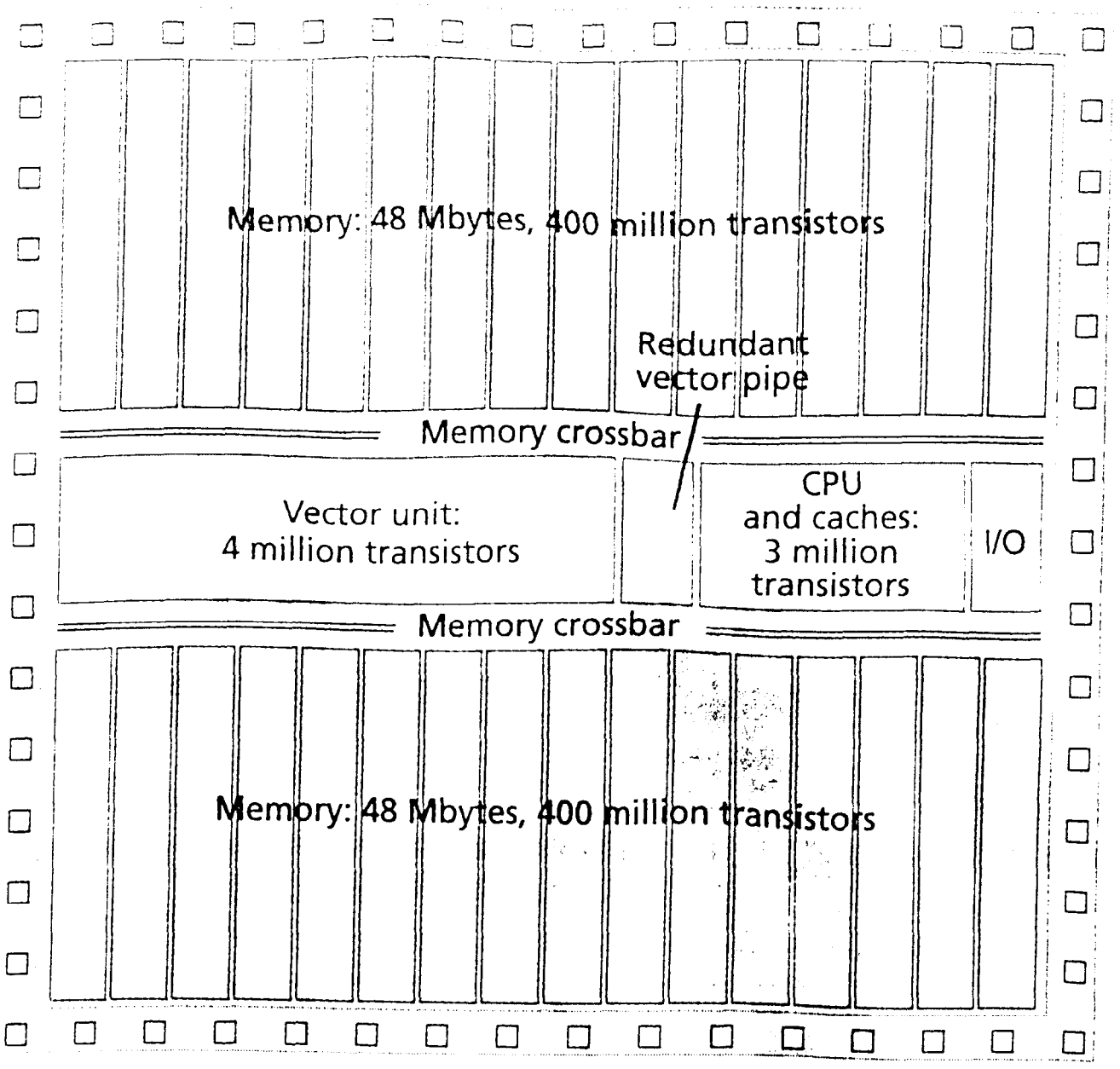
C. CMP

6 CMP T.1, p.80

Table 1. Characteristics of superscalar, simultaneous multithreading, and chip multiprocessor architectures.

Characteristic	Superscalar	Simultaneous multithreading	Chip multiprocessor
Number of CPUs	1	1	8
CPU issue width	12	12	2 per CPU
Number of threads	1	8	1 per CPU
Architecture registers (for integer and floating point)	32	32 per thread	32 per CPU
Physical registers (for integer and floating point)	32 + 256	256 + 256	32 + 32 per CPU
Instruction window size	256	256	32 per CPU
Branch predictor table size (entries)	32,768	32,768	8 × 4,096
Return stack size	64 entries	64 entries	8 × 8 entries
Instruction (I) and data (D) cache organization	1 × 8 banks	1 × 8 banks	1 bank
I and D cache sizes	128 Kbytes	128 Kbytes	16 Kbytes per CPU
I and D cache associativities	4-way	4-way	4-way
I and D cache line sizes (bytes)	32	32	32
I and D cache access times (cycles)	2	2	1
Secondary cache organization (Mbytes)	1 × 8 banks	1 × 8 banks	1 × 8 banks
Secondary cache size (bytes)	8	8	8
Secondary cache associativity	4-way	4-way	4-way
Secondary cache line size (bytes)	32	32	32
Secondary cache access time (cycles)	5	5	7
Secondary cache occupancy per access (cycles)	1	1	1
Memory organization (no. of banks)	4	4	4
Memory access time (cycles)	50	50	50
Memory occupancy per access (cycles)	13	13	13

IRAM Fig. 1, p. 77



D. IRAM

7 Raw Fig. 1, p. 87

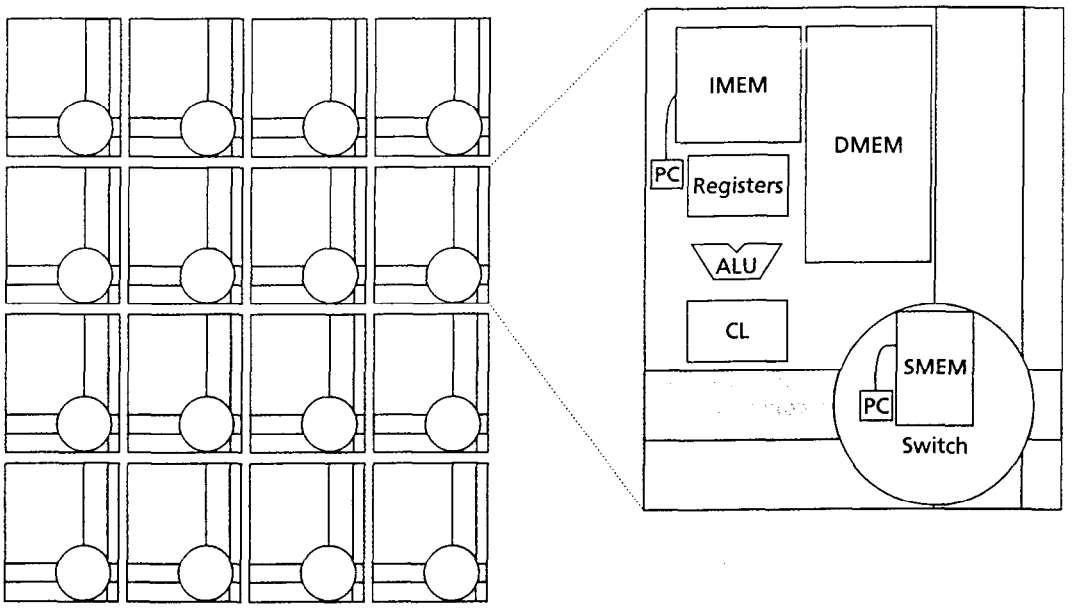


Figure 1. A Raw processor is constructed of multiple identical tiles. Each tile contains instruction memory (IMEM), data memories (DMEM), an arithmetic logic unit (ALU), registers, configurable logic (CL), and a programmable switch with its associated instruction memory (SMEM).

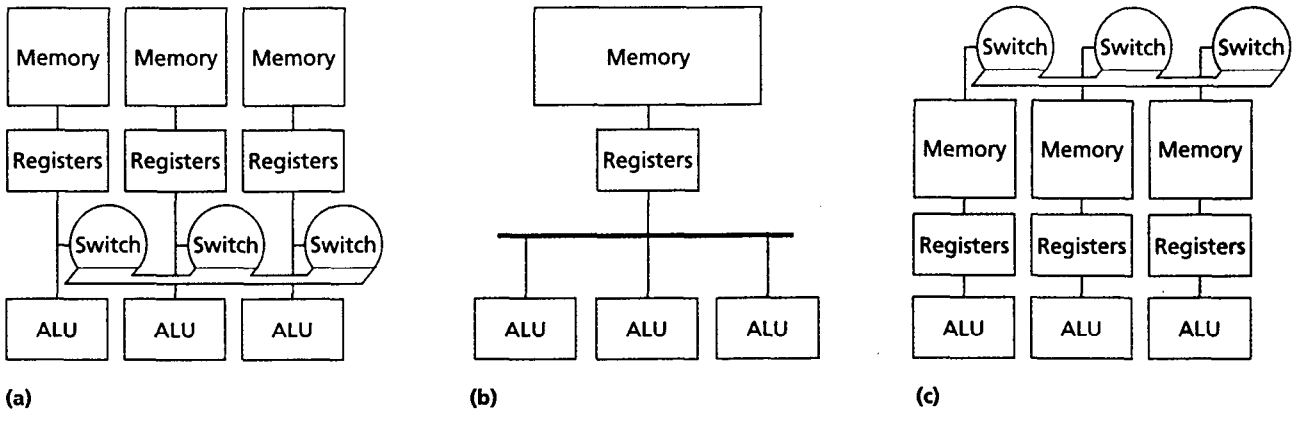


Figure 2. Raw microprocessors differ from superscalar and multiprocessor architectures. (a) Raw microprocessors distribute the register file and memory ports and communicate between ALUs on a switched, point-to-point interconnect. (b) A superscalar contains a single register file and memory port and communicates between ALUs on a global bus. (c) Multiprocessors communicate at a much coarser grain through the memory subsystem.

E. RAW A. SSS C. CMP

p. 87

MAIN PRINCIPLES OF EPIC:

1.THE COMPILER SHOULD PLAY THE KEY ROLE IN DESIGNING THE PLAN OF EXECUTION (POE), AND THE ARCHITECTURE SHOULD PROVIDE THE REQUISITE SUPPORT FIR IT TO DO SO SUCCESSFULLY.

2.THE ARCHITECTURE SHOULD PROVIDE FEATURES THAT ASSIST THE COMPILER IN EXPLOITING STATISTICAL ILP.

3.THE ARCHITECTURE SHOULD PROVIDE MECHANISMS TO COMMUNICATE THE COMPILER'S POE TO THE HARDWARE.

Table 1.1. Play Doh features and the dynamic capabilities they are intended to replace.

PlayDoh Feature	**Dynamic Capability Replaced**
MultiOp instructions	Multiple instruction issue / dynamic parallel dependence analysis
Static resource allocation / explicit specification of resource allocation	Dynamic parallel resource allocation
Static scheduling	Out-of-order execution
Static register renaming / rotating registers	Dynamic register renaming
Static branch prediction Predicated execution	Dynamic branch prediction
Prepare-to-branch instruction	Branch target buffer
Speculative code motion / speculative opcodes / exception tags Predicated execution	Speculative execution
Static disambiguation / static scheduling Statistical disambiguation / data speculation	Out-of-order memory referencing

ADVANTAGES OF EPIC:

1.SINCE THE COMPILER DOES MOST OF THE WORK - LESS AND SIMPLER HARDWARE

2.SCHEDULING DONE ONLY ONCE AT COMPILE TIME. SAVE COMPUTING TIME

3.OPTIMIZING AT COMPILE TIME BENEFITS THE PROGRAM EVERY TIME IT IS RUN

PREDICATION

PREDICATE - CONDITION TELLING THE INSTRUCTION WHEN TO EXECUTE AND WHEN NOT.

REDUCES THE NUMBER OF CONDITIONAL BRANCHES.

THE CONDITION HAS TO BE KNOWN BY THE TIME THE CONDITIONAL INSTRUCTIONS HAVE TO BE RETIRED.

THE PREDICATE FIELD (6 BITS) SELECTS ONE SET OUT OF 64 1-BIT PREDICATE REGISTERS.

A PREDICATE REGISTER IS SET TO 1 (TRUE) IF THE CONDITION IS TRUE AND TO 0 (FALSE) IF THE CONDITION IS FALSE.

SIMULTANEOUSLY AND AUTOMATICALLY, IT SETS ANOTHER PREDICATE REGISTER TO THE INVERSE VALUE.

USING REDICATION, THE MACHINE LANGUAGE INSTRUCTION FORMING THE THEN AND ELSE CLAUSES WILL BE MERGED INTO A SINGLE STREAM OF INSTRUCTIONS, THE FORMER ONES USING THE PREDICATE AND THE LATTER USING ITS INVERSE.

EXAMPLE:

(a)HLL if STATEMENT:

```
if (R1 = R2) then R3 = R4 + R5
else R6 = R4 - R5
```

(b)ASSEMBLY CODE FOR (a):

```
    CMP R1,R2;   R1 = R2 ?
    BNE L1;      branch on non-equal to L1
    MOV R3,R4;   R3 <--- R4
    ADD R3,R5;   R3 <--- R4 + R5
    BR  L2;      unconditionally branch to L2
L1: MOV R6,R4;   R6 <--- R4
    SUB R6,R5;   R6 <--- R4 - R5
L2:
```

(c)PREDICATED EXECUTION:

```
    CMPEQ  R1,R2,P4
    <P4> ADD  R3,R4,R5
    <P5> SUB  R6,R4,R5
```

THE CMPEQ INSTRUCTION COMPARES R1 AND R2 AND SETS THE PREDICATE REGISTER P4 TO 1 IF THEY ARE EQUAL AND TO 0 IF NOT EQUAL.

IT ALSO SETS PREDICATE REGISTER P5 TO THE INVERSE CONDITION.

IF THE CONDITION IS SATISFIED (R1 = R2), THEN ONLY ADD WILL BE EXECUTED. OTHERWISE, ONLY SUB. BOTH ARE ACTUALLY EXECUTED, BUT ONLY THE CORRECT ONE IS RETIRED (COMMITTED).

Figure 2-1. System Environments

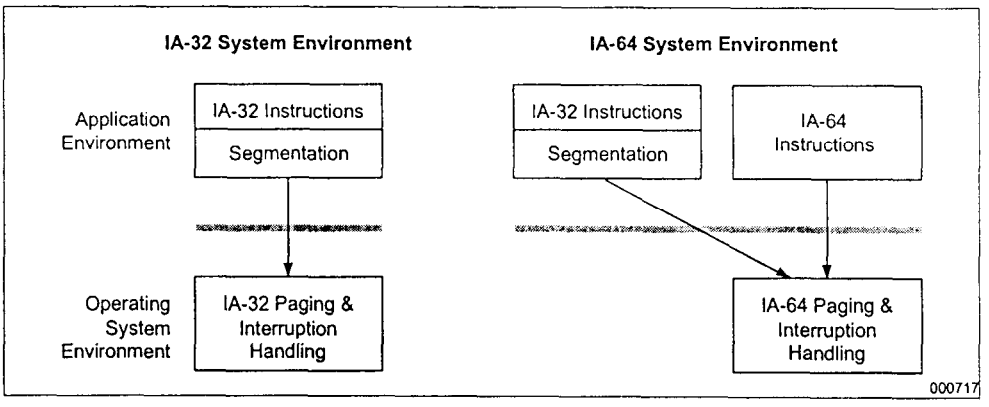
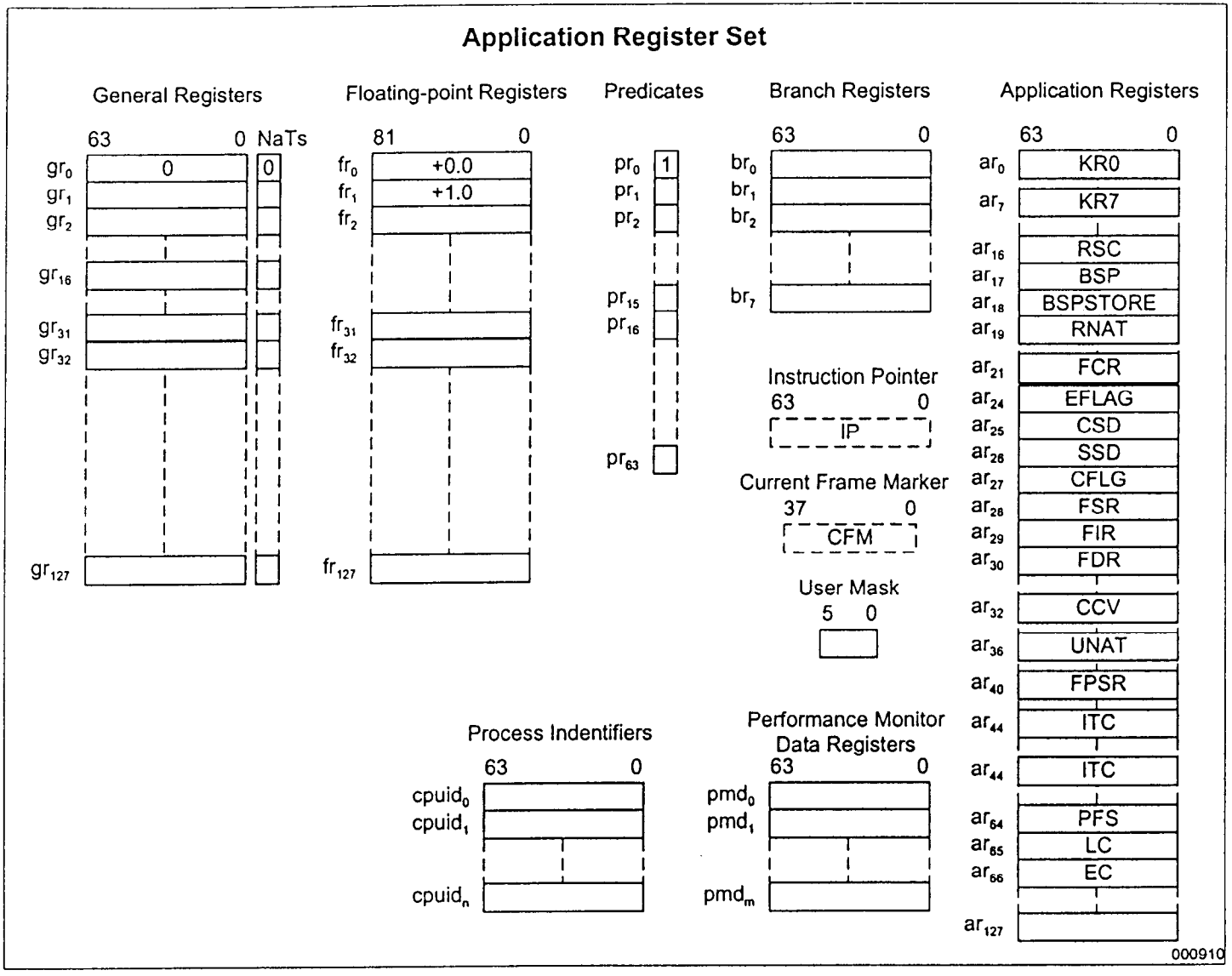


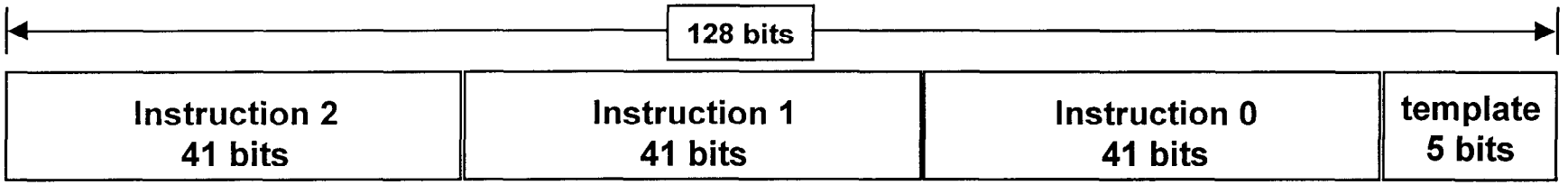
Table 2-1. Major Operating Environments for IA-64 Processors

System Environment	Application Environment	Usage
IA-32 System Environment	IA-32 Instruction Set	IA-32 PM, RM and VM86 application and operating system environment. Compatible with IA-32 Pentium®, Pentium Pro, Pentium II and Pentium III processors.
	IA-64 Instruction Set	Not supported, IA-64 applications cannot execute in the IA-32 system environment.
IA-64 System Environment	IA-32 Protected Mode	IA-32 Protected Mode applications in the IA-64 system environment.
	IA-32 Real Mode	IA-32 Real Mode applications in the IA-64 system environment.
	IA-32 Virtual Mode	IA-32 Virtual 86 Mode applications in the IA-64 system environment.
	IA-64 Instruction Set	IA-64 Applications on IA-64 operating systems.

Figure 3-1. Application Register Model



Instruction Formats: Bundles



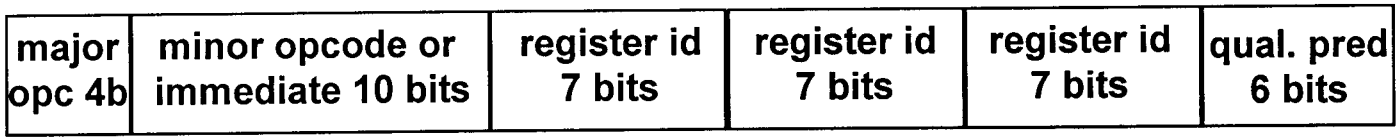
Instruction Types

- M: Memory
- I: Shifts, MM
- A: ALU
- B: Branch
- F: Floating point
- L+X: Long

Template types

- Regular: MII, MLX, MMI, MFI, MMF
- Stop: MI_I M_MI
- Branch: MIB, MMB, MFB, MBB, BBB
- All come in two versions:
  - with stop at end
  - without stop at end

Instruction Formats: Instructions



Qualifying predicates (6 bits)

- A few instructions do not have a QP

Register operand identifiers (7 bits)

Register result identifier(s) (6 or 7 bits)

Immediate operands (8-22 bits)

Minor opcode

Major opcode (4 bits)

3.1.7 Current Frame Marker

Each general register stack frame is associated with a frame marker. The frame marker describes the state of the IA-64 general register stack. The Current Frame Marker (CFM) holds the state of the current stack frame. The CFM cannot be directly read or written (see "Register Stack" on page 4-1).

The frame markers contain the sizes of the various portions of the stack frame, plus three Register Rename Base values (used in register rotation). The layout of the frame markers is shown in Figure 3-2 and the fields are described in Table 3-2.

On a call, the CFM is copied to the Previous Frame Marker field in the Previous Function State register (see Section 3.1.8.10). A new value is written to the CFM, creating a new stack frame with no locals or rotating registers, but with a set of output registers which are the caller's output registers. Additionally, all Register Rename Base registers (RRBs) are set to 0. See "Modulo-scheduled Loop Support" on page 4-28.

Figure 3-2. Frame Marker Format

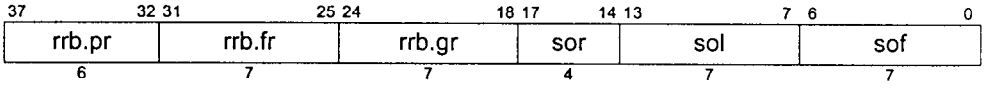


Table 3-2. Frame Marker Field Description

Field	Bit Range	Description
sof	6:0	Size of stack frame
sol	13:7	Size of locals portion of stack frame
sor	17:14	Size of rotating portion of stack frame (the number of rotating registers is 8 * sor)
rrb.gr	24:18	Register Rename Base for general registers
rrb.fr	31:25	Register Rename Base for floating-point registers
rrb.pr	37:32	Register Rename Base for predicate registers

Figure 4-1. Register Stack Behavior on Procedure Call and Return

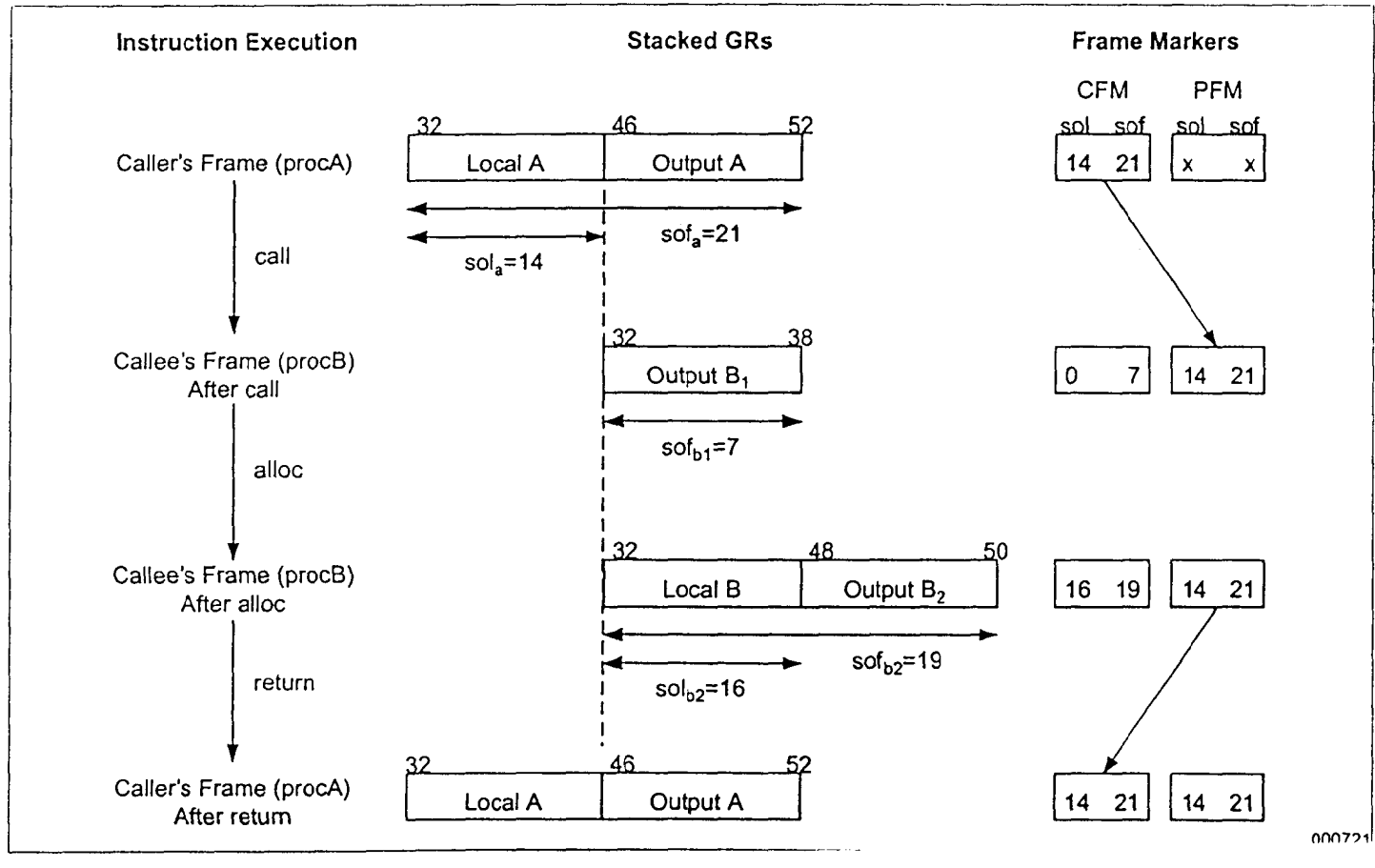
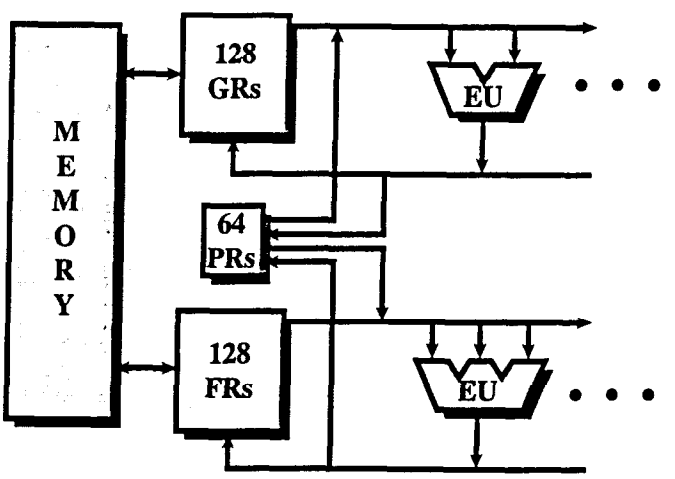


Table 13.2 Traditional Superscalar versus IA-64 Architecture

Superscalar	IA-64
RISC-line instructions, one per word	RISC-line instructions bundled into groups of three
Multiple parallel execution units	Multiple parallel execution units
Reorders and optimizes instruction stream at run time	Reorders and optimizes instruction stream at compile time
Branch prediction with speculative execution of one path	Speculative execution along both paths of a branch
Loads data from memory only when needed, and tries to find the data in the caches first	Speculatively loads data before its needed, and still tries to find data in the caches first



GR = General-purpose or integer register
FR = Floating-point or graphics register
PR = One-bit predicate register
EU = Execution unit

Figure 13.18 General Organization for IA-64 Architecture.

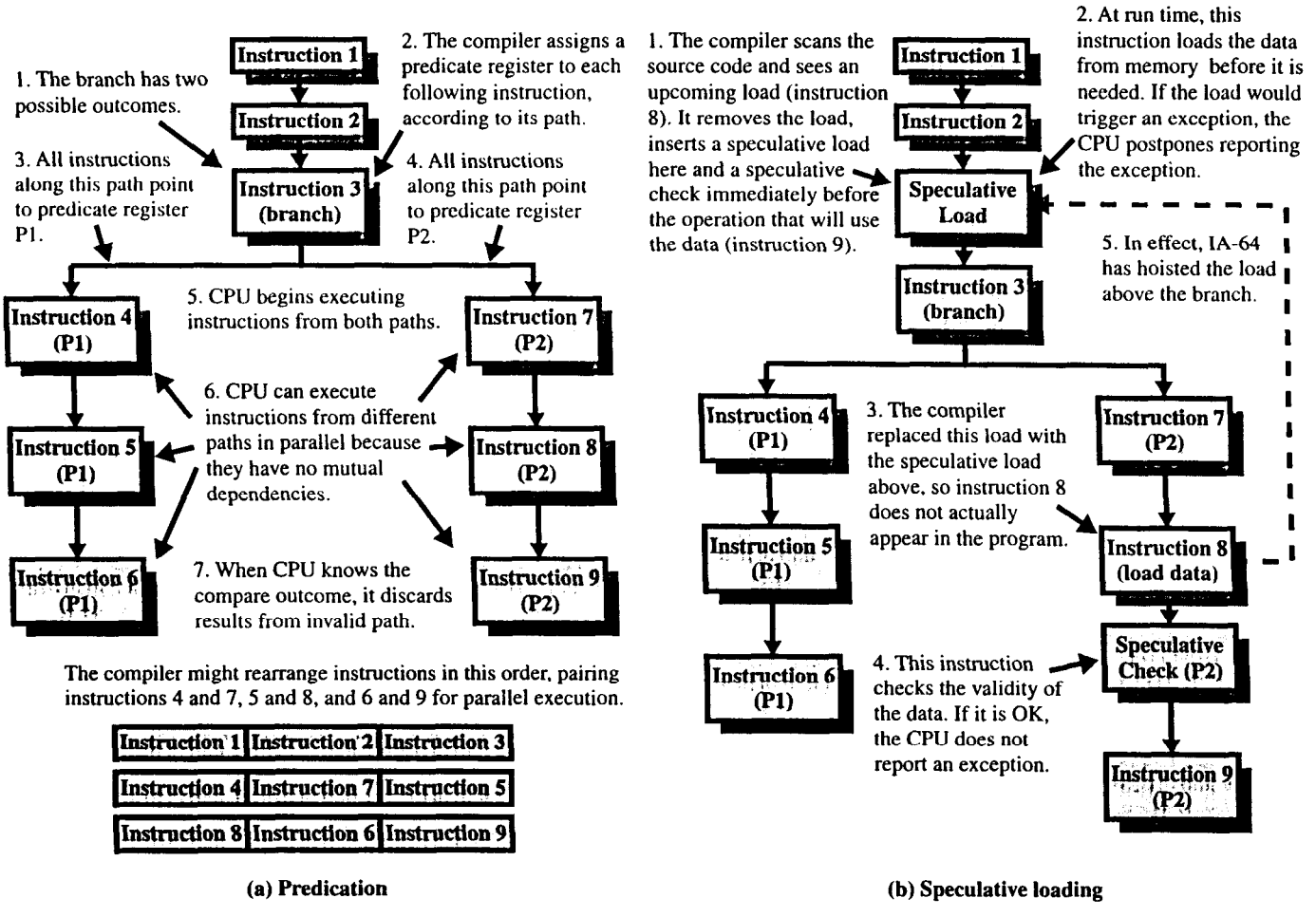


Figure 13.20 IA-64 Predication and Speculative Loading.

DIFFICULTIES WITH IA-64:

(1) SUCH AN ADVANCED SYSTEM - NEVER BUILT.
EXPECT PITFALLS.

(2) IA-64 SMART COMPILERS HAVE TO BE WRITTEN.

(3) OS MUST BE FULLY 64-BIT.
WILL HAVE TO SWITCH FROM WINDOWS 95,98
TO NT OR UNIX-TYPE.

(4) FULL COMPATIBILITY WITH INTEL IA-32 AND
HP PA-RISC MAY CAUSE PROBLEMS.

(5) THERE MAY BE COMPETING ALTERNATIVES (SUCH AS
ALPHA) FROM OTHER VENDORS; CONVENTIONAL RISC
SYSTEMS OF HIGHER PERFORMANCE.