
APPENDIX I

The PowerPC

For many years, the personal computer market was dominated by two different CPU architectures: the Intel 80×86 series and the Motorola 680×0 series of microprocessors. In 1981, IBM introduced the first PC based on Intel 8088 architecture (an 8086 with an 8-bit data bus). Clone PC manufacturers quickly stepped in and far surpassed IBM as suppliers of PCs. In 1984, Apple Computer began using the 680×0 architecture in its line of Macintosh computers, but saw its market share shrink as the Intel-based PCs took over the market. In order to stay competitive, Apple, IBM, and Motorola (AIM) formed an alliance in 1991 to develop a new microprocessor. This processor, called the PowerPC (Power Performance Computing), would be designed using RISC, not CISC, techniques. IBM already used RISC methods in their RS/6000-series workstations and servers and adapted their multi-chip POWER (Performance Optimization With Enhanced RISC) architecture when developing the PowerPC in order to tap into a large base of existing RS/6000 software. Motorola's design work on their second-generation 88110 RISC CPU also influenced the initial PowerPC design.

The first PowerPC CPU to hit the market was the PowerPC 601, introduced in 1993. More CPUs followed, as indicated in Table I.1. Each new PowerPC CPU generation provided faster clock speeds and improved architectural features, while at the same time consuming less power than that of an equivalent Intel processor. IBM and Motorola now offer many different PowerPC CPUs, including versions designed for the embedded controller market.

The PowerPC supports three types of code applications: Conventional, Native, and FAT. These are described as follows:

- Conventional: Application contains older 680×0 code which is emulated by the PowerPC
- Native: Application contains PowerPC code
- FAT: Application contains both types of code, 680×0 and Native

TABLE I.1 PowerPC CPU generation timeline

<i>Generation</i>	G1	G2	G3	G4	G5
<i>Year</i>	1993	1994	1997	1999	2003
<i>Model</i>	601	603/603e 604/604e	740/745 750/755	7400/7410 7440/7450	85xx
<i>Clock Speed (MHz)</i>	33-120	200-375	233-700	450-1500	1600-2400
<i>Feature(s)</i>	RISC architecture superscalar	Separate data and instruction cache	Backside L2 cache controller	AltiVec vector processor engine	Velocity engine Full 64-bit data paths/registers

Emulated code runs slower than Native code (as Native code is used to simulate the operation of the emulated code). However, as the PowerPC's clock speed increased, emulated 680×0 code began to run faster than the actual 680×0 processor that the code was written for.

INSIDE THE POWERPC: SOFTWARE

Table I.2 shows a summary of the PowerPC instruction set. As with the Intel 80×86 architecture, the PowerPC diverges slightly from the pure RISC path by offering a large instruction set. In keeping with the RISC philosophy, the PowerPC instructions rely heavily on the use of register-to-register operations. This can be seen in the following example PowerPC code:

```
back:
    lbz     r6, 10(r8)
    add    r4, r4, r6
    stb    r4, 20(r8)
    not    r4, r4
    stb    r4, 24(r8)
    srwi   r4, r4, 8
    stb    r4, 28(r20)
    subi   r7, r7, 1
    cmpwi  r7, 0
    bne    back
    blr
```

The PowerPC provides 32 general-purpose registers (r0 through r31) and 32 floating-point registers (f0 through f31), as well as a program counter, link register (for handling subroutines), condition register, and other control/status registers. Note that an instruction may contain up to three operands (two source operands and one destination operand). To take full advantage of the PowerPC architecture, an optimizing compiler should be used to generate the assembly language.

TABLE I.2 Simplified PowerPC: CPU instruction set

<i>Instruction Mnemonics</i>	<i>Description</i>
add / add. / addo / addo.	Add
addc / addc. / addco / addco.	Add Carrying
adde / adde. / addeo / addeo.	Add Extended
addi / li / la / subi	Add Immediate
addic / subic	Add Immediate Carrying
addic. / subic.	Add Immediate Carrying and Record
addis / lis / subis	Add Immediate Shifted
addme / addme. / addmeo / addmeo.	Add to Minus One Extended
addze / addze. / addzeo / addzeo.	Add to Zero Extended
and / and.	AND
andc / andc.	AND with Complement
andi.	AND Immediate
andis.	AND Immediate Shifted
b / ba / bl / bla	Branch
bc / bca / bcl / bcla	Branch Conditional
bcctr / bcctrl	Branch Conditional to Count Register
bclr / bclrl	Branch Conditional to Link Register
cmp / cmpw	Compare
cmpi / cmpwi	Compare Immediate
cmpl / cmplw	Compare Logical
cmpli / cmplwi	Compare Logical Immediate
cntlzw / cntlzw.	Count Leading Zeros Word
crand	Condition Register AND
crandc	Condition Register AND with Complement
creqv / crset	Condition Register Equivalent
crnand	Condition Register NAND
crnor / crnot	Condition Register NOR
cror / crmove	Condition Register OR
crorc	Condition Register OR with Complement
crxor / crclr	Condition Register XOR
divw / divw. / divwo / divwo.	Divide Word
divwu / divwu. / divwuo / divwuo.	Divide Word Unsigned
eqv / eqv.	Equivalent
extsb / extsb.	Extend Sign Byte
extsh / extsh.	Extend Sign Half Word
lbz	Load Byte and Zero
lbzu	Load Byte and Zero with Update
lbzux	Load Byte and Zero with Update Indexed

(continued on next page)

TABLE I.2 (continued)

<i>Instruction Mnemonics</i>	<i>Description</i>
lbzx	Load Byte and Zero Indexed
lha	Load Half Word Algebraic
lhau	Load Half Word Algebraic with Update
lhaux	Load Half Word Algebraic with Update Indexed
lhax	Load Half Word Algebraic Indexed
lhbrx	Load Half Word Byte-Reverse Indexed
lhz	Load Half Word and Zero
lhzu	Load Half Word and Zero with Update
lhzux	Load Half Word and Zero with Update Indexed
lhzx	Load Half Word and Zero Indexed
lmw	Load Multiple Word
lswi	Load String Word Immediate
lswx	Load String Word Indexed
lwbrx	Load Word Byte-Reversed Indexed
lwz	Load Word and Zero
lwzu	Load Word and Zero with Update
lwzux	Load Word and Zero with Update Indexed
lwzx	Load Word and Zero Indexed
mcrf	Move Condition Register Field
mcrxr	Move to Condition Register from XER
mfer	Move from Condition Register
mfspr / mfxer / mflr / mfctr	Move from Special-Purpose Register
mftb	Move from Time Base
mterf	Move to Condition Register Fields
mtspr / mtixer / mtlr / mtctr	Move to Special-Purpose Register
mulhw / mulhw.	Multiply High Word
mulhwu / mulhwu.	Multiply High Word Unsigned
mulli	Multiply Low Immediate
mullw / mullw. / mullwo / mullwo.	Multiply Low Word
nand / nand.	NAND
neg / neg. / nego / nego.	Negate
nor / nor.	NOR
or / or.	OR
orc / orc.	OR with Complement
ori	OR Immediate
oris	OR Immediate Shifted
rlwimi / rlwimi. / inslwi / insrwi	Rotate Left Word Immediate then Mask Insert

TABLE I.2 (continued)

<i>Instruction Mnemonics</i>	<i>Description</i>
<i>rlwinm / rlwinm. / extlwi / extrwi rotlwi / rotrwi / slwi / srwi clrlwi / clrrwi / clrlslwi</i>	Rotate Left Word Immediate then AND with Mask
<i>rlwnm / rlwnm. / rotlw</i>	Rotate Left Word then AND with Mask
<i>slw / slw.</i>	Shift Left Word
<i>sraw / sraw.</i>	Shift Right Algebraic Word
<i>srawi / srawi.</i>	Shift Right Algebraic Word Immediate
<i>srw / srw.</i>	Shift Right Word
<i>stb</i>	Store Byte
<i>stbu</i>	Store Byte with Update
<i>stbux</i>	Store Byte with Update Indexed
<i>stbx</i>	Store Byte Indexed
<i>sth</i>	Store Half Word
<i>sthbrx</i>	Store Half Word Byte-Reverse Indexed
<i>sthu</i>	Store Half Word with Update
<i>sthux</i>	Store Half Word with Update Indexed
<i>sthx</i>	Store Half Word Indexed
<i>stmw</i>	Store Multiple Word
<i>stswi</i>	Store String Word Immediate
<i>stswx</i>	Store String Word Indexed
<i>stw</i>	Store Word
<i>stwbrx</i>	Store Word Byte-Reverse Indexed
<i>stwu</i>	Store Word with Update
<i>stwux</i>	Store Word with Update Indexed
<i>stwx</i>	Store Word Indexed
<i>subf / subf. / subfo / subfo. / sub</i>	Subtract From
<i>subfc / subfc. / subfco / subfco. / subc</i>	Subtract from Carrying
<i>subfe / subfe. / subfeo. / subfeo.</i>	Subtract from Extended
<i>subfic</i>	Subtract from Immediate Carrying
<i>subfme / subfme. / subfmeo / subfmeo.</i>	Subtract from Minus One Extended
<i>subfze / subfze. / subfzeo / subfzeo.</i>	Subtract from Zero Extended
<i>xor / xor.</i>	XOR
<i>xori</i>	XOR Immediate
<i>xoris</i>	XOR Immediate Shifted

INSIDE THE POWERPC: HARDWARE

Let us take a quick look at the internal architecture of several PowerPC CPUs. Figure I.1 illustrates the PowerPC 601 block diagram. This is the first PowerPC CPU developed by the AIM alliance. The Branch Prediction Unit (BPU), Integer Unit (IU), and Floating Point Unit (FPU) are all fed instructions simultaneously, a characteristic of superscalar RISC designs. An on-chip instruction/data cache helps reduce accesses to main memory.

In Figure I.2, we see the second-generation (G2) improvements made in the PowerPC 603 processor. The integrated instruction/data cache of the 601 processor has been split into separate instruction and data caches, each with its own memory management unit (D MMU and I MMU). A Completion Unit handles proper completion of out-of-order instructions.

The third-generation (G3) PowerPC 750 CPU, illustrated in Figure I.3, contains additional improvements. A second Integer Unit has been added, along with Reservation Stations to assist with instruction scheduling. A level-2 (L2) backside cache controller with its own address and data bus has also been added.

The fourth-generation (G4) PowerPC contains an important new hardware component: the AltiVec vector processing engine. To speed up multimedia, encryption, compression, and many other data-intensive applications, the AltiVec engine processes data in 128-bit chunks using 162 new SIMD (Single Instruction Multiple Data) instructions.

The latest processor, the PowerPC G5, offers a full 64-bit architecture, with a 64-bit address bus, a 64-bit data bus, and 64-bit registers. Based on IBM's 64-bit POWER4 architecture, the PowerPC G5 is heavily pipelined, allowing up to 215 in-flight instructions. Dual Floating-Point Units, dual Integer Units, dual Load/Store Units, and the Velocity Engine vector processing unit provide a highly parallel processing environment. A redesigned frontside bus allows data to travel into and out of the processor simultaneously at a maximum speed of 8 GB/sec. Addressable memory jumps from 4 GB in the PowerPC G4 to 4 Terabytes (4096 GB). The PowerPC G5 also contains larger L1 and L2 caches (32-KB L1 data cache, 64-KB L1 instruction cache, and 512-KB L2 cache). All code written for the 32-bit PowerPC G4 runs without change on the 64-bit PowerPC G5, including the 32-bit Mac OS X operating system. The PowerPC G5 competes favorably with Intel's Pentium 4 CPU.

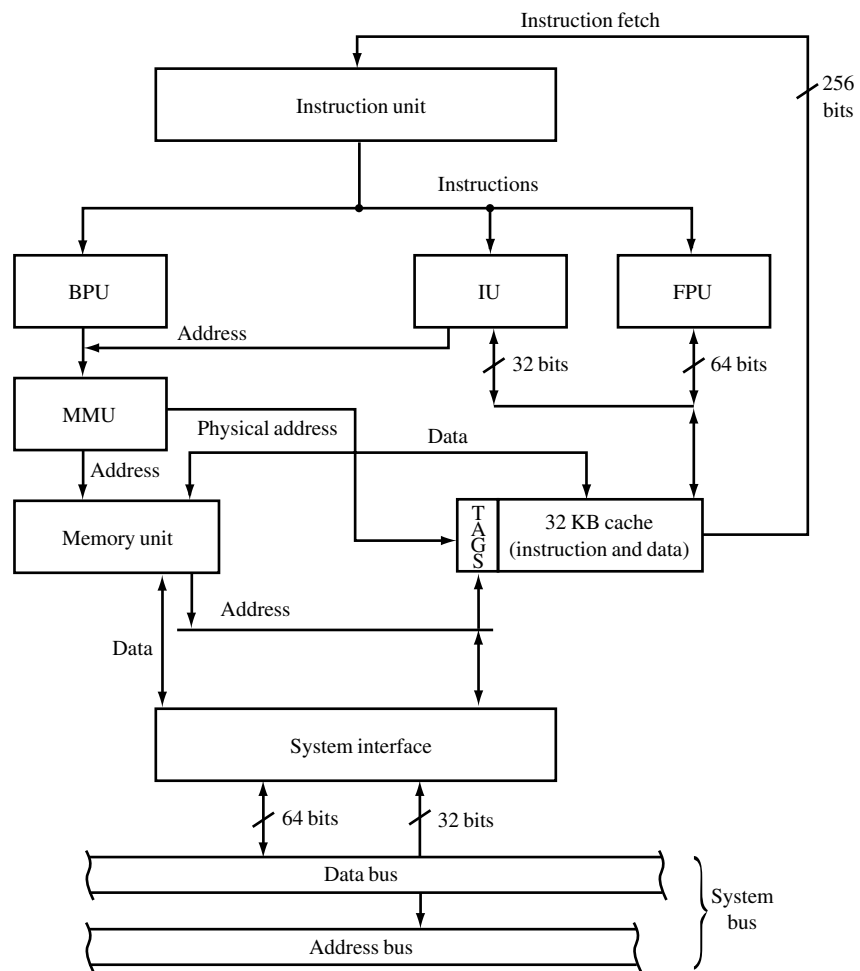


FIGURE I.1 The PowerPC 601 CPU block diagram

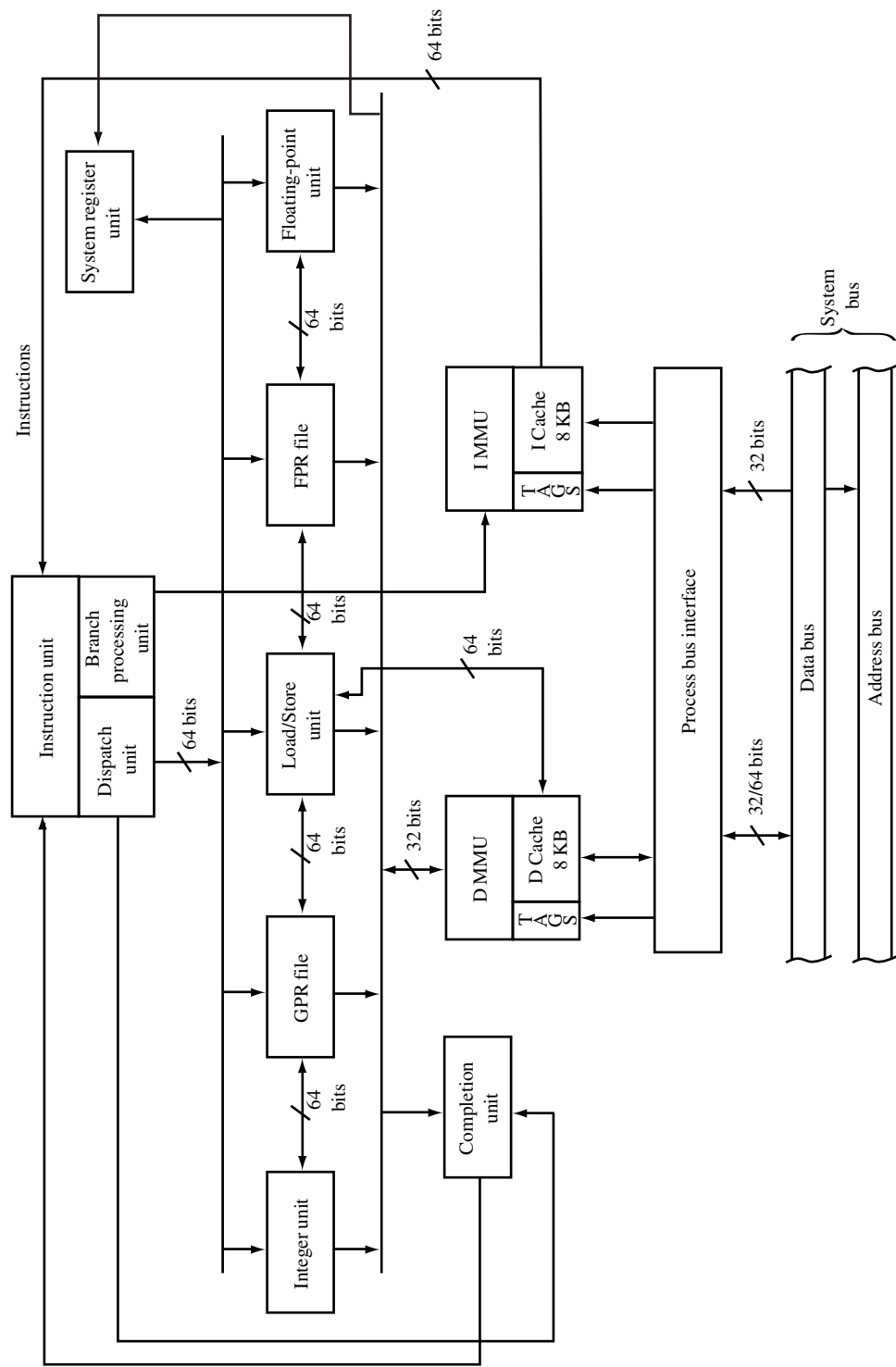


FIGURE I.2 The PowerPC 603 CPU block diagram

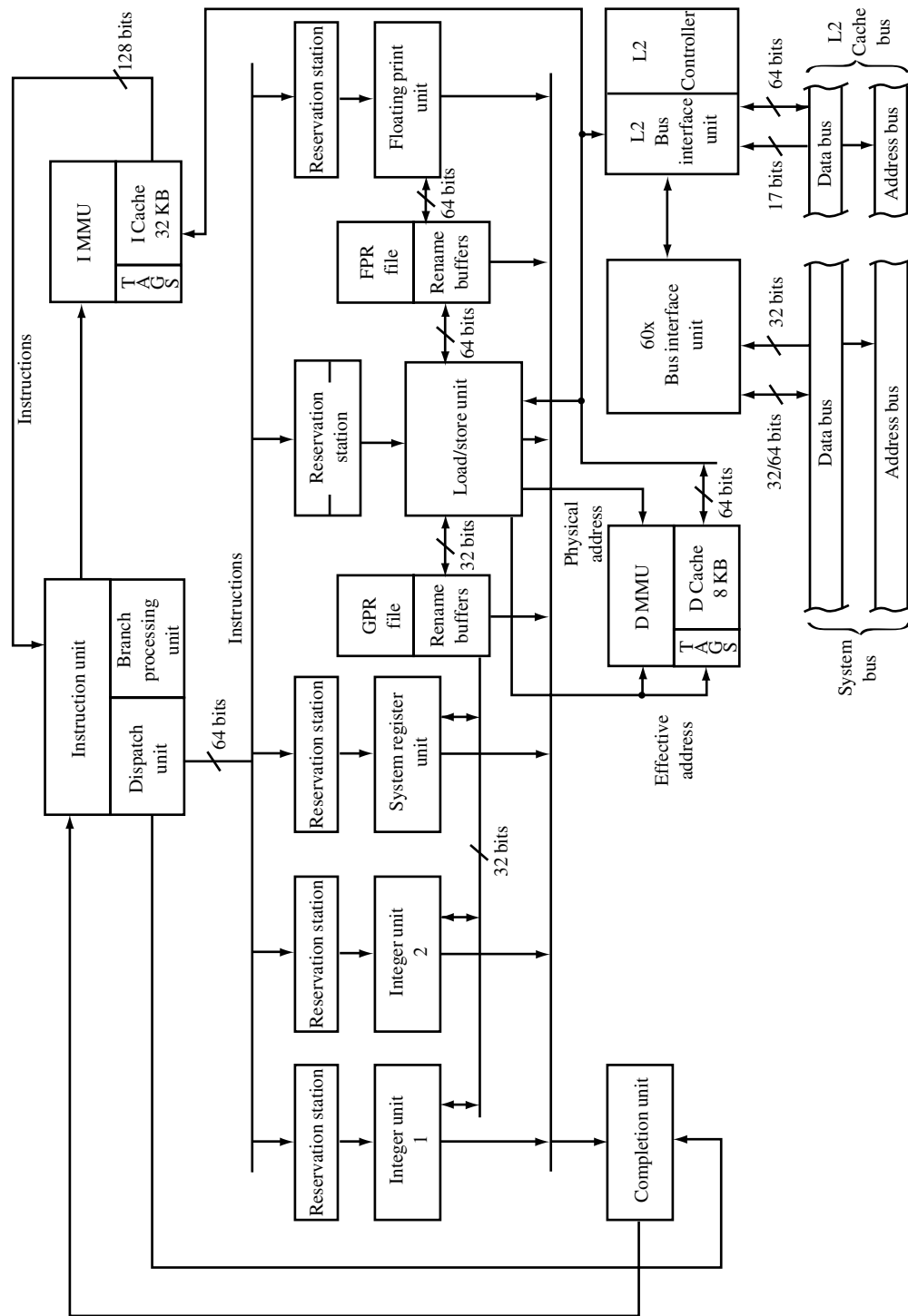


FIGURE I.3 The PowerPC 750 CPU block diagram

THE FUTURE?

What is next for the PowerPC? Information from Motorola suggests that the sixth-generation PowerPC will contain a 256-bit AltiVec II vector processing unit (versus 128 bits for AltiVec on the G4 and G5) and run at clock speeds from 4 to 10 GHz. The seventh-generation PowerPC may reach a clock speed of 20 GHz.

In the past 10 years, the PowerPC has evolved into a significant processing architecture, well suited for the performance demands of multimedia. Who knows where the future will take the PowerPC?

For additional information on the PowerPC, visit the following Web sites:

- <http://www.apple.com/g5processor>
- <http://www-3.ibm.com/chips/products/powerpc>
- <http://www.motorola.com/powerpc>