MIPS - Is it important? [closed]

Asked 16 years, 3 months ago Modified 2 years, 5 months ago

Viewed 14k times

28

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 2 years ago.

Improve this question

My question: Is the MIPS programming language that beneficial to know?

I'm a CS student and am taking an assembly class which focuses on MIPS. I'm very comfortable writing in high level languages, but MIPS has me a little bit down.

Is MIPS something that I should really focus on and try to completely grasp? Will it help me in future?

assembly mips

edited Jul 12, 2022 at 18:30

Sep Roland

38.7k ● 9 ● 47 ● 86

asked Sep 19, 2008 at 2:58

user17083

415 ● 1 ● 6 ● 8

14 Answers

Sorted by: Highest score (default)

At one point (in the 90s) MIPS-derived processors were the best selling processors in the world, dwarfing sales of Intel x86 processors. This was because of their huge presence in the embedded market. I think now ARM-based processors may have taken over that title, but there are still tons of embedded systems out there using MIPS.

Even if you never program a MIPS chip in assembler in your career, assembly language can be useful to learn. It can help you write more efficient high level code if you have some idea of what the compiler is going to emit. Other areas where it is still used include compilers (writing your own), device drivers, and multimedia programming (where code requiring MMX or SSE is usually still written by hand in assembler).

Each CPU type has a different instruction set but there's enough commonality that once you learn one dialect of assembly (MIPS in your case) the others should be easy to pick up.

Share Improve this answer Follow

edited Sep 19, 2008 at 3:23

answered Sep 19, 2008 at 3:07

sk.

6,406 ● 5 ● 40 ● 47

10

It is helpful to understand the very low levels of the computer. For example there are developers who will say the CPU executes some sort of "new object" instruction. There is of course no such instruction, management of objects happens several layers of abstraction higher. It is good to understand the distinction, so you can understand why object creation might be expensive (or not) or why protected addresses spaces can make the system more robust.

When I was in school, the assembly course was taught on the IBM mainframe using the System/360 instruction set. I have never, at any point in my career, come anywhere near working on such a machine, but the knowledge of what the CPU looks like has been valuable.

Nowadays I work on embedded systems using MIPS processors. I actually spend a reasonable amount of time poring over MIPS assembly listings, and write some MIPS assembly for the boot vectors and synchronization primitives. Yet even if you never actually write anything in assembly, understanding CPU operation is still valuable.

Share Improve this answer

Follow

answered Sep 19, 2008 at 3:12

DGentry

16.3k ● 8 ● 53 ● 66

8

I took on a assembly class doing mips about two years ago. I found myself writing GameBoy Advance games in a mips-like asm language. Can't say I enjoyed it.

Though I think it is good to have an understanding about the inner workings of assembled code (from high level to low level). I feel it made me understand more about how the computer actually works. Also, I couldn't resist making a few virtual machines shortly after, designing my own assembly language :)

Share Improve this answer

Follow

answered Sep 19, 2008 at 3:01

Statement

4,038 ● 3 ● 40 ● 45

6

MIPS is a great language to learn assembly with. You have plenty of general purpose registers to make writing your programs less tedious and it is a RISC architecture so there are less instructions you need to memorize.

Some of the mainstream usage that not a lot of people mention is that the N64, Playstation 1, Playstation 2 and PSP all used MIPS processors.

Share Improve this answer

Follow

edited Jun 26, 2009 at 11:16

answered May 4, 2009 at 9:38

mmcdole

92.7k ● 61 ● 188 ● 224

4

By "mips programming language" I assume you mean MIPS Assembly Language.

Learning an assembly language is beneficial to know, it helps you map the code you write to how it will run on the hardware. This can help expose you to more low level concerns like cache misses, branching, out-of-order execution, and other things you wouldn't think about writing high level code.

Weather learning MIPS assembly is useful to you depends on what career you plan to follow. Personally I think x86/x64 (popular in PCs, directly applicable if you write C++ or C# on PC), PPC (gaining popularity, used in game consoles, and also exposes you to RISC) or ARM (used in a lot of embedded devices such as mobile phones) might be better choices to learn.

Share Improve this answer

Follow

answered Sep 19, 2008 at 3:13

redwyre

1,099 ● 10 ● 24

4

MIPS specifically is less important than understanding asm in general so you have some idea of what happens when you compile high-level source. Understanding a bit

of asm can help you make sense of the symptoms of bugs in high-level code. (e.g. overwriting the wrong data through a bogus pointer, or modifying other locals on the stack when you write outside an array).

Obviously understanding how to make C or C++ run fast is much easier when you know that what *actually* runs is the compiler-generate asm. It's impossible to design good microbenchmarks to test anything if you don't already mostly know what's going on.

Once you've learned MIPS assembly, it will be pretty easy to learn any other. All other mainstream CPUs are also register machines that work basically the same way:

- There are registers and memory
- Everything including pointers are just bits and bytes in memory or registers, i.e. integers.
- Each instruction is independent, and updates the architectural state (register values) according to its simple rules. It only cares about its inputs (usually registers, sometimes also memory). There is no magic.
- Execution continues from one instruction to the next in increasing address in program memory, except for jump / branch instructions.
- Writing programs / functions is a matter of **constructing a series of steps for the machine to take** that will result in data being moved around and

processed the way you want. It's like designing the steps in an algorithm

- ASM is where a debugger really shines: it can show you the full architectural state of the machine (because there are a finite number of registers), and show you which register values changed when you single-step by one instruction.

Every architecture has its own extra wrinkles, but these basics don't change and are what you really learn when you learn your first assembly language. One of MIPS's major wrinkle is the branch-delay slot (which the MARS and SPIM simulators hide / disable by default)

RISC-V is a lot like MIPS, but without quirks like a branch-delay slot. And there is a RARS simulator with a toy system-call environment a lot like MARS provides for MIPS. As a bonus, RISC-V is a lot more commercially-relevant these days, so knowing it specifically has more real-world benefit if it catches your interest. But you can still imagine a classic-RISC pipeline that implements RISC-V, and learn all the same computer-architecture basics.

MIPS is a pretty nice assembly language to learn. It's simple and orthogonal, and leads nicely to discussions of pipelined CPUs because that's what it was designed for. (No microcoded instructions, and very regular machine-code format that's easy to decode.)

Also, there are nice MIPS simulators, MARS and SPIM, which have an editor / assembler / simulator + debugger all in one. And some "system calls" which do high-level things like read an integer from the user's keyboard into a register. Normal OSes have system calls that just let you read/write characters, and you have to call library functions or write your own integer->string functions. This is a blessing and a curse: if you don't realize that MARS system calls are basically library functions with a special calling convention, you might not realize that you can write your own code that does some of those things, or that it's not "normal" for system calls to work this way.

You could learn all this on x86, though, especially if you want to do performance experiments on your desktop / laptop.

Share Improve this answer Follow

edited Jul 13, 2022 at 5:38

answered Apr 12, 2018 at 3:51

Peter Cordes
361k • 49 • 699 • 958

3

Assembly language is always worth learning. You probably aren't being taught or seeing how the compiler turns high-level language code to assembly code, but I'd say that knowledge is the most important benefit of knowing assembly language. It's sobering how inefficient many compilers are even on the highest optimization

level. (Most compilers will dump assembly code for you on request, instead of going on to machine code and linking.)

That said, MIPS asm is probably useless, because almost nobody uses MIPS now. (SGI used to, but now their machines are all on x86/IA-64 chips.) If you find assembly language appealing and want to work in assembly further, learn the ARM instruction set. ARM is only a little more complex than MIPS, and virtually all mobile phones, smart phones, and PDAs now use ARM chips (made by dozens of manufacturers). The iPod uses ARM, as does (I think) the Zune.

Share Improve this answer

Follow

answered Sep 19, 2008 at 3:16

Michael Ratanapintha

40.4k ● 4 ● 34 ● 42

3

Here are 3 good reasons for you to learn assembly.

1. Certain small pieces of programs are better written in assembly than in a high level language.
2. Old assembly programs may need to be debugged or enhanced.
3. Learning to program in assembly will help you to develop precise models for what a processor does and what a compiler does.

The third reason is by far the most important. Knowing how your hardware works will greatly deepen your

understanding of computer science and in turn help you to become a better developer.

Share Improve this answer

Follow

answered Feb 17, 2015 at 23:02

JeLLyB0x3r

328 ● 2 ● 9

2

Mips appear to be coming back in embedded systems. @[sk] has given a better summary than me, but I have a recent example :

Broadcom appear to make Wifi cards these days with either MIPS or ARM chips on them (I'm not sure which tho, documentation says one, firmware says the other). However, on thier newer chips we're still awaiting on working linux support with all features enabled, so for that reason, I'm glad you're learning. (Now get and write me some drivers :P)

Share Improve this answer

Follow

answered Sep 19, 2008 at 3:16

Kent Fredric

57.3k ● 14 ● 111 ● 151

2

In your case, I think MIPS is used as a tutoring aid. This is a computer science class and rather than give you code monkey tools, they are trying to teach you broad concepts that are easier exemplified on a simpler architecture like MIPS rather than on x86.

Do not expect it to enter your "programmer's toolbox". For the practical purpose of writing low level code, you'd better learn x86 assembly language afterwards.

Share Improve this answer

Follow

answered Jan 9, 2012 at 0:08

Ilya

5,603 ● 2 ● 31 ● 57

1

MIPS Assembly Language is definitely a nice language to learn compared to other assembly languages, but it's not useful in the job market concerning finding a job.

If you want to do something for yourself—then yes—it's pretty nice. But besides that, not really.

Share Improve this answer

Follow

edited Jul 16, 2022 at 8:18

jackdbd

5,005 ● 4 ● 30 ● 41

answered Jul 12, 2022 at 17:21

CrustyCrustecean

30 ● 5

1

I would invest some time into it during the semester. The reason why they teach MIPS in school is because it is a reduced instruction set and less complex than the processor currently in your computer. In addition, the company was founded by an academic professor with background in the industry. The CEO of MIPS wrote one of the more well-known books on computer architecture.

Outside of the course, you will gain more benefit from understanding how the ALU, Controller, Registers, etc. work and less from the actual programming language. In my opinion, the course is really about Computer Architecture but taught through MIPS as a medium due to its relative simplicity.

Share Improve this answer

Follow

edited Jul 24, 2022 at 16:18

tripleee

189k ● 36 ● 311 ● 359

answered Jul 24, 2022 at 15:42

A P

2,528 ● 3 ● 29 ● 39

1

I have studied MIPS last year. MIPS is an important language, but unfortunately, you will probably never use MIPS. You will learn some basics, but don't waste time trying to learn more than you need, because there are not to many jobs in MIPS, compared to other languages.

Share Improve this answer

Follow

edited Jul 24, 2022 at 16:19

tripleee

189k ● 36 ● 311 ● 359

answered Oct 21, 2013 at 17:36

user2868657

I think it depends what area in CS you want to shoot towards after graduation. If computer architecture is what

0

you want to do, then in my opinion I say it is. I think a good software engineer walks away with the general concept of how an assembly language works + bypassing and forwarding and finally how your code can be optimized for cache performance. (Spatial locality/temporal locality)

Share Improve this answer

Follow

answered Apr 30, 2014 at 18:24

user3124081

412 ● 1 ● 4 ● 16

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.