

**pi@raspberrypi:~/tests$ as -o hello.o hello.s**

hello.s: Assembler messages:  
hello.s:1: Error: unknown architecture armv7-a@Target' hello.s:2: Error: bad expression hello.s:2: Error: junk at end of line, first unrecognized character isC'  
hello.s:3: Error: junk at end of line, first unrecognized character is @' hello.s:6: Error: junk at end of line, first unrecognized character is@'  
hello.s:7: Error: operand 1 must be an integer register -- mov r7,#4@\_\_NR\_write=4 on 32-bit ARM EABI[7][3]' hello.s:8: Error: operand 1 must be an integer register --mov r0,#1@fd=1(stdout)[7]'  
hello.s:9: Error: operand 1 must be an integer register -- ldr r1,=msg@buf=&msg[7]' hello.s:10: Error: operand 1 must be an integer register --mov r2,#12@count=12 bytes("Hello World\n")[5]'  
hello.s:11: Error: unknown mnemonic swi' --swi #0@trap to kernel(svc)[13]'  
hello.s:13: Error: junk at end of line, first unrecognized character is @' hello.s:14: Error: operand 1 must be an integer register --mov r7,#1@\_\_NR\_exit=1 on 32-bit ARM EABI[7][3]'  
hello.s:15: Error: operand 1 must be an integer register -- mov r0,#0@status=0[7]' hello.s:16: Error: unknown mnemonicswi' -- swi #0@trap to kernel(svc)[13]' hello.s:18: Error: bad expression hello.s:18: Error: junk at end of line, first unrecognized character isD'  
hello.s:20: Error: junk at end of line, first unrecognized character is `@'

Those errors come from two issues: using invalid comment text in the source and mixing assembler syntaxes/mnemonics that GAS doesn’t accept as written on the Pi. Use GAS-friendly syntax, valid comments, and either SVC or SWI with unified syntax. Below is a minimal, working 32-bit ARMv7 Linux “Hello World” for a Raspberry Pi OS 32‑bit userland.

**Minimal source**

* Save as hello.s, assemble and link with: as -o hello.o hello.s && ld -o hello hello && ./hello.[[1]](#fn1)[[2]](#fn2)

.syntax unified @ Use ARM’s unified syntax so SVC/SWI are recognized [^1][^8]  
 .text @ Code section [^5]  
 .global \_start @ Make \_start visible to linker [^3]  
  
\_start:  
 @ write(1, msg, len) with EABI: r7=syscall#, r0..r2=args [^4]  
 mov r0, #1 @ fd = 1 (stdout) [^5]  
 ldr r1, =msg @ buf = &msg [^5]  
 mov r2, #12 @ count = 12 ("Hello World\n") [^3]  
 mov r7, #4 @ \_\_NR\_write = 4 on 32-bit ARM EABI [^4]  
 svc #0 @ make syscall (SWI/SVC are equivalent) [^1][^8]  
  
 @ exit(0) [^4]  
 mov r0, #0 @ status = 0 [^5]  
 mov r7, #1 @ \_\_NR\_exit = 1 [^4]  
 svc #0 @ syscall [^1]  
  
 .data @ Data section [^5]  
msg:  
 .ascii "Hello World\n" @ 12 bytes [^3]

**Why the previous errors**

* “unknown architecture `armv7-a@Target’” came from including text after the directive; remove “@Target …” and use clean directives like .syntax unified and .text without trailing prose. GAS treats “@” as a comment in ARM mode, but only after a valid token; random trailing text can still confuse parsing if attached to a directive.[[3]](#fn3)[[4]](#fn4)
* “unknown mnemonic swi” occurs if not in unified syntax or if the assembler defaults to a mode where SVC is the accepted mnemonic; enabling .syntax unified and using svc #0 fixes it, and SWI and SVC are equivalent mnemonics historically.[[5]](#fn5)[[6]](#fn6)
* “operand 1 must be an integer register” errors were caused by comments jammed onto the same token (e.g., “#4@\_\_NR\_write…” merged); keep a space before comments so the immediate stays “#4” and comment text follows.[[3]](#fn3)[[2]](#fn2)

**Environment notes**

* This program targets 32-bit ARM EABI userland on Pi; it uses r7 for the syscall number and svc #0 to enter the kernel, with args in r0–r2 for write and return value in r0, per Linux-on-ARM EABI convention.[[7]](#fn7)[[8]](#fn8)
* On a 64-bit Raspberry Pi OS (aarch64), conventions differ (x0–x2 args, x8 syscall number, write=64, exit=93) and this exact file won’t run; use a 32-bit OS/userland or port the code to aarch64 conventions.[[8]](#fn8)[[9]](#fn9)

⁂

1. <https://embeddedjourneys.com/blog/hello-world-arm-assembly-raspberry-pi/>

1. <https://kevinboone.me/pi-asm-01_exit.html>

1. <https://stackoverflow.com/questions/63769229/telling-gas-to-use-semicolons-for-comments-in-intel-syntax-mode>

1. <https://peterdn.com/post/2019/02/03/hello-world-in-arm-assembly/>

1. <https://stackoverflow.com/questions/8459279/are-arm-instructuons-swi-and-svc-exactly-same-thing>

1. <https://developer.arm.com/documentation/ddi0406/cb/Application-Level-Architecture/Instruction-Details/Alphabetical-list-of-instructions/SVC--previously-SWI-?lang=en>

1. <https://thinkingeek.com/2014/05/24/arm-assembler-raspberry-pi-chapter-19/>

1. <https://manpages.debian.org/testing/manpages-dev/syscall.2.en.html>

1. <https://chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md>

1. <https://man7.org/linux/man-pages/man2/syscall.2.html>

1. <https://stackoverflow.com/questions/51543818/att-syntax-hello-world-works-but-intel-syntax-does-not>

1. <https://keleshev.com/compiling-to-assembly-from-scratch/07-arm-assembly-programming>

1. <https://stackoverflow.com/questions/76567156/how-does-arm-svc-instruction-works>

1. <https://stackoverflow.com/questions/12946958/what-is-the-interface-for-arm-system-calls-and-where-is-it-defined-in-the-linux>

1. <https://www.reddit.com/r/raspberry_pi/comments/147pd7/are_any_of_you_using_your_raspberry_pi_to_learn/>

1. <https://dercuano.github.io/notes/arm-assembly.html>

1. <https://dev.iachieved.it/iachievedit/working-with-arm-assembly/>

1. <https://gist.github.com/yamnikov-oleg/454f48c3c45b735631f2>

1. <https://www.ics.com/blog/get-know-arm-assembly-language>

1. <https://www.utdallas.edu/~pervin/RPiA/RPiA.pdf>