CSUN | CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP 122

DR JEFF SOFTWARE
INDIE APP DEVELOPER

COMP122

8-27-23

ASSEMBLY Programming

Lab Basics

Dr Jeff Drobman

website → drjeffsoftware.com/classroom.html

email → jeffrey.drobman@csun.edu

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

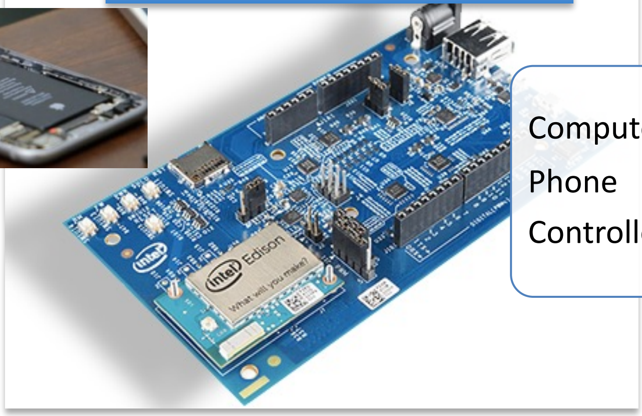
Index

DR JEFF SOFTWARE
INDIE APP DEVELOPER

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

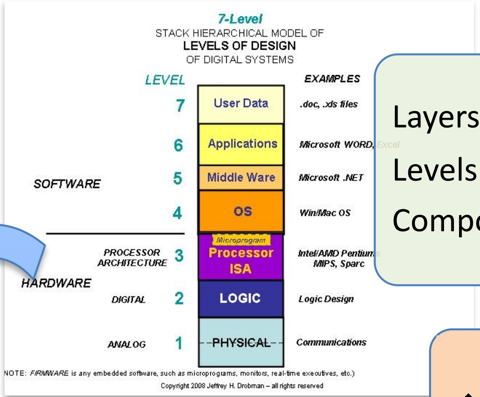
Digital Systems

Hardware



Computer
Phone
Controller

Software


7-Level
STACK HIERARCHICAL MODEL OF
LEVELS OF DESIGN
OF DIGITAL SYSTEMS
LEVEL
EXAMPLES
7 User Data .doc, .xls files
6 Applications Microsoft WORD, Excel
5 Middle Ware Microsoft .NET
4 OS Win/Mac OS
SOFTWARE
HARDWARE
PROCESSOR ARCHITECTURE 3 Microprogram Processor ISA Intel/AMD Pentiums MIPS, Sparc
DIGITAL 2 LOGIC Logic Design
ANALOG 1 PHYSICAL Communications
NOTE: FIRMWARE is any embedded software, such as microprograms, monitors, real-time executives, etc.)
Copyright 2008 Jeffrey H. Drobman – all rights reserved

Layers
Levels
Composition

- ❖ CODE
- ❖ DATA

Computer **Engineering**

- ❖ **Architecture**
- ❖ Physical design

External Storage

Hard disk
Flash disk
Flash
EEPROM

Computer **Science**

- ❖ **Algorithms**
- ❖ Theory

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

Lab Programs

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023

1. "Hello World": I/O in MIPS & ARM
2. Number systems and radix conversion
3. BCD on LED
4. Moving data (memory <-> GR< -> FPU <-> CP0)
5. "Hello World" extended: loops, macros, functions/subroutines
6. Computation 1: Fibonacci (add, overflow)
7. Computation 2: Factorials (mult, overflow)
8. Bit-wise operations (bit masks, shifts); example: tic-tac-toe
9. Interrupt/Exception handler

10. Project 1: LED (MMIO, delay loops, speed slider)
11. Project 2: ISA design (logic design & sim new instructions)

2	3	3	3	3	3	3	5	5	7	8
LAB	LAB	LAB	LAB	LAB	LAB	LAB	LAB	LAB	Proj	Proj
1	2	3	4	5	6	7	8	9	1	2

Section

Key Slides

- ❖Instruction Formats/Classes
- ❖Numbers
- ❖ASCII Codes

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

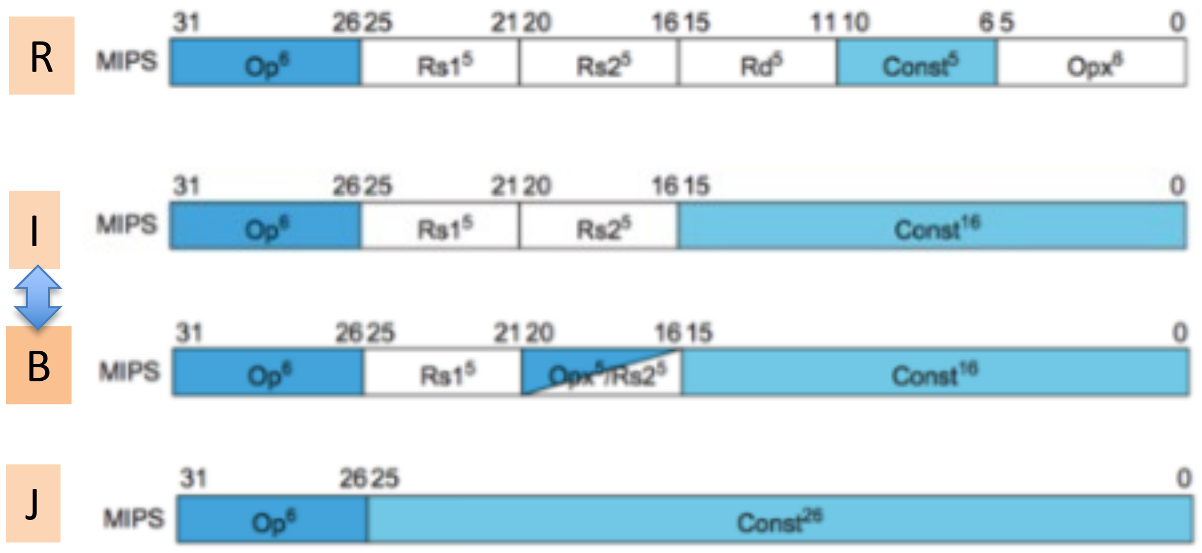
Instruction Formats

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023

```
                   6        5        5         5        5        6
               +--------+--------+--------+--------+--------+---------+
R-type format  | Op-code|   Rs   |   Rt   |   Rd   |   SA   |Funct-code|
               +--------+--------+--------+--------+--------+---------+

                   6        5        5                 16
               +--------+--------+--------+--------------------------+
I-type format  |Op-code |     Rs |     Rt | 2's complement constant  |
               +--------+--------+--------+--------------------------+

                   6                        26
               +--------+--------------------------------------------+
J-type format  | Op-code|                jump_target                 |
               +--------+--------------------------------------------+
                ^                                                    ^
```



CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

Baseline Instruction Set

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023

Rev Jan 2021

Computation

- ALU
  - ADD
  - SUB
  - AND
  - OR
  - XOR
  - NOT
- MULT/DIV [opt]
- BIT
  - SET/CLR
  - TEST
- COMPARE
  - CMP
- SHIFT
  - SHIFT (A, L)
  - ROTATE

Memory

- Reg-Reg
  - MOV
- Reg-Mem
  - LOAD
  - STORE
  - MOV
- Mem-Mem
  - MOV
- Stack
  - PUSH
  - POP

Program Control

- JUMP
  - JUMP/GOTO
- BRANCH
  - BRA
  - BRCC
  - LOOP
- CALL
  - CALL/CALR/JAL
  - RET/RETFIE
- NOP

I/O

- I/O
  - IN
  - OUT
- Mem Mapped
  - MOV PORT
  - LOAD/STORE

System Control

- Reset
  - RESET
- Power
  - SLEEP/HALT

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

Ordinals

```
Technical ordinals
-----------
10^(-24)  yacto
10^(-21)  zepto
10^(-18)  atto
10^(-15)  femto
10^(-12)  pico
10^(-9)  nano
10^(-6)  micro
10^(-3)  milli
10^(-2)  centi
10^(-1)  deci
10^(+1)  deka
10^(+2)  hecto
10^(+3)/2^(10)   kilo
10^(+6)/2^(20)   mega
10^(+9)/2^(30)   giga
10^(+12)/2^(40)   tera
10^(+15)/2^(50)   peta
10^(+18)/2^(60)   exa
10^(+21)/2^(70)   zetta
10^(+24)/2^(80)   yotta
```

10^(29)/2^(100) **geo**

Gazillions

10^(+6)	million
10^(+9)	billion
10^(+12)	trillion
10^(+15)	quadrillion
10^(+18)	quintillion
10^(+21)	sexillion
10^(+24)	septillion
10^(+27)	octillion
10^(+30)	nonillion
10^(+33)	decillion
10^(+36)	undecillion
10^(+39)	duodecillion
10^(+42)	tredecillion
10^(+45)	quattuordecillion
10^(+48)	quindecillion
10^(+51)	sexdecillion
10^(+54)	septendecillion
10^(+57)	octodecillion
10^(+60)	novemdecillion
10^(+63)	vigintillion
10^(+100)	googol
10^(+303)	centillion
10^(10^(+100))	googolplex

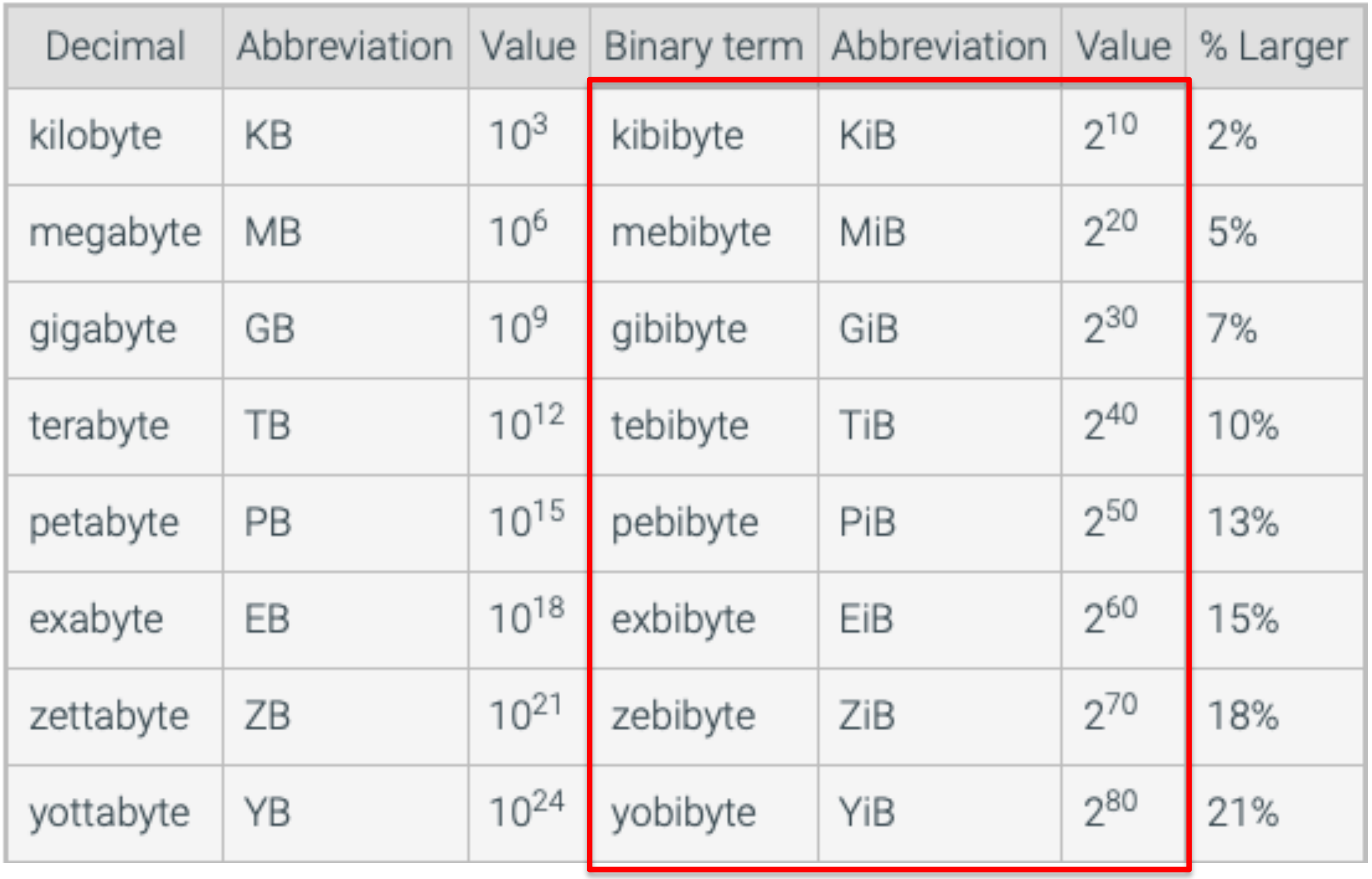
Ordinal	Power of 2	Power of 10	Actual
1K	2^{10}	10^3	1024
1M	2^{20}	10^6	1,048,576
1G	2^{30}	10^9	$1.074\text{x}10^9$
1T	2^{40}	10^{12}	$1.0995\text{x}10^{12}$

Name	2^n	M/G	Actual
byte	2^8	--	256
short	2^{16}	64K	65,536
word	2^{32}	4B	$4.3\text{x}10^9$
long	2^{64}	16 Q	$1.84\text{x}10^{19}$
IPv6	2^{128}	340 uD	$3.4\text{x}10^{38}$

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

GiB/TiB Ordinals

DR JEFF SOFTWARE
INDIE APP DEVELOPER



Decimal	Abbreviation	Value	Binary term	Abbreviation	Value	% Larger
kilobyte	KB	10^3	kibibyte	KiB	2^{10}	2%
megabyte	MB	10^6	mebibyte	MiB	2^{20}	5%
gigabyte	GB	10^9	gibibyte	GiB	2^{30}	7%
terabyte	TB	10^{12}	tebibyte	TiB	2^{40}	10%
petabyte	PB	10^{15}	pebibyte	PiB	2^{50}	13%
exabyte	EB	10^{18}	exbibyte	EiB	2^{60}	15%
zettabyte	ZB	10^{21}	zebibyte	ZiB	2^{70}	18%
yottabyte	YB	10^{24}	yobibyte	YiB	2^{80}	21%

Actual
1024
1,048,576
1.074×10^9
1.0995×10^{12}

COMP122

ASCII Codes- Letters

1963

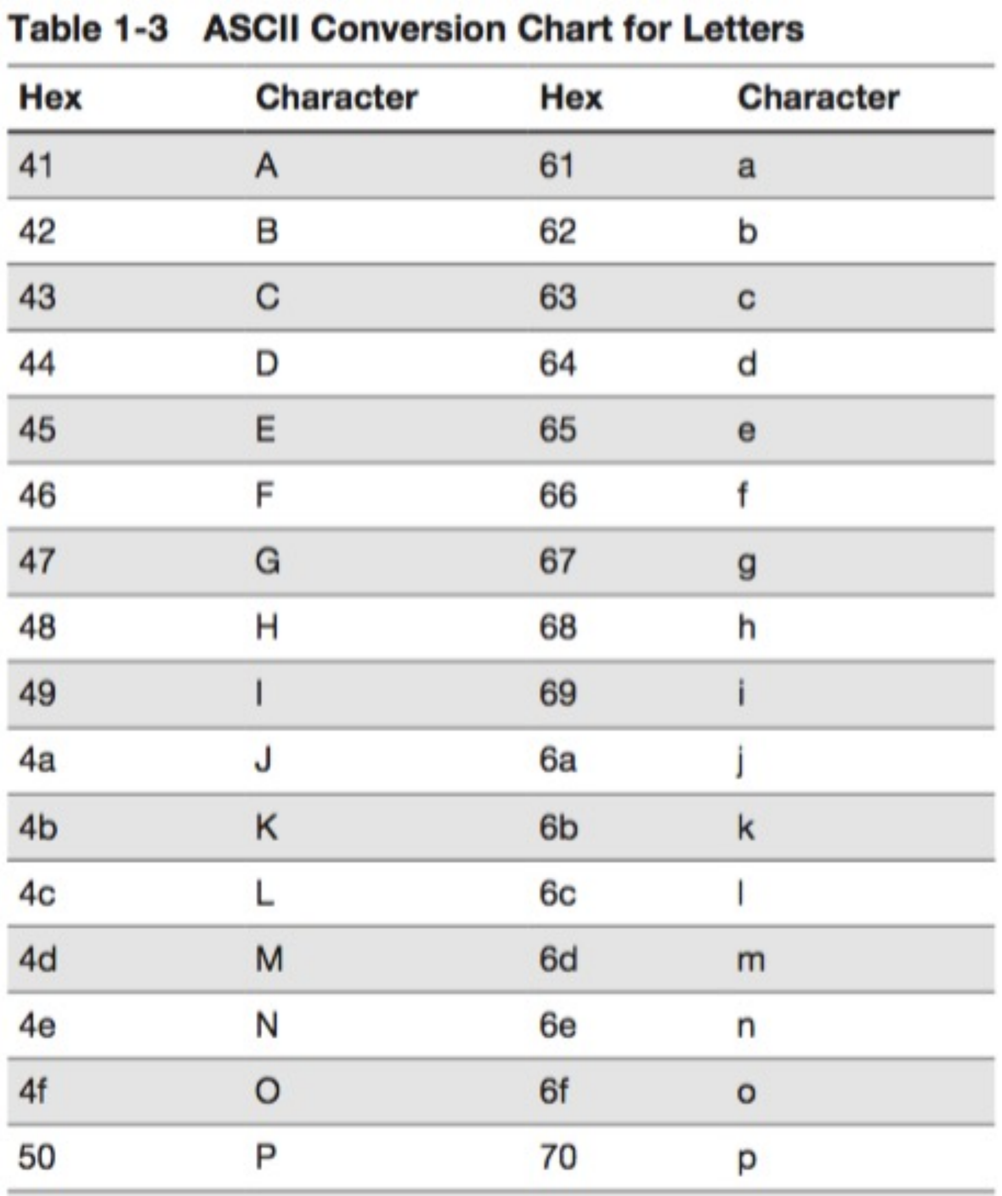


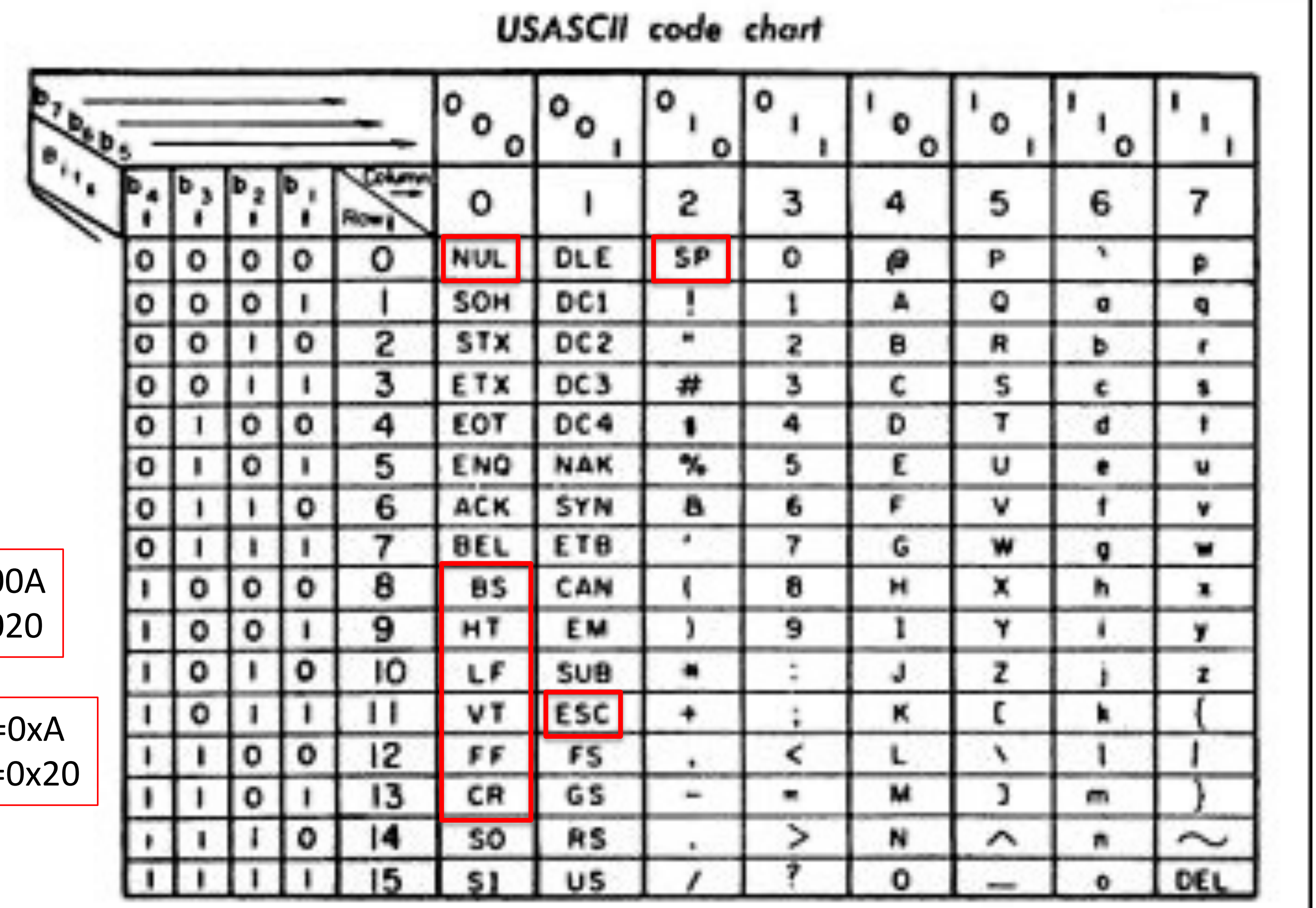
Table 1-3 ASCII Conversion Chart for Letters

Hex	Character	Hex	Character
41	A	61	a
42	B	62	b
43	C	63	c
44	D	64	d
45	E	65	e
46	F	66	f
47	G	67	g
48	H	68	h
49	I	69	i
4a	J	6a	j
4b	K	6b	k
4c	L	6c	l
4d	M	6d	m
4e	N	6e	n
4f	O	6f	o
50	P	70	p

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

ASCII Codes- 7-bit

DR JEFF SOFTWARE
INDIE APP DEVELOPER



USASCII code chart

b7 b6 b5 →					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b4	b3	b2	b1	Column → / Row ↓	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

\n=\u000A
sp=\u0020

char ch=0xA
char sp=0x20

IANA encourages use of the name "US-ASCII" for Internet uses of ASCII

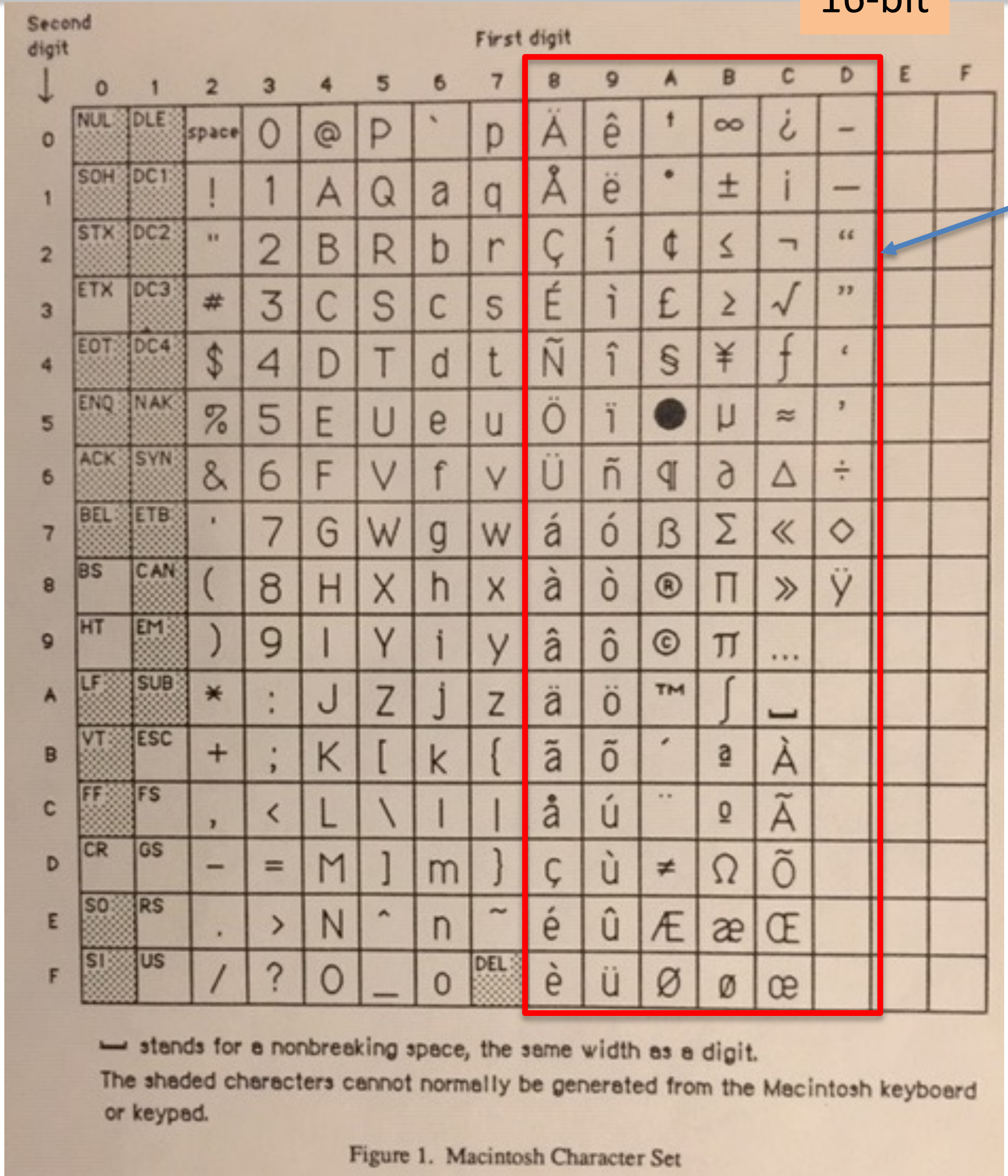
CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

Old Mac Char Codes

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023

16-bit

unique
special chars



First digit

Second digit ↓	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DLE	space	0	@	P	`	p	Ä	ê	†	∞	¿	–		
1	SOH	DC1	!	1	A	Q	a	q	Å	ë	°	±	¡	—		
2	STX	DC2	"	2	B	R	b	r	Ç	í	¢	≤	¬	“		
3	ETX	DC3	#	3	C	S	c	s	É	ì	£	≥	√	”		
4	EOT	DC4	$	4	D	T	d	t	Ñ	î	§	¥	ƒ	‘		
5	ENQ	NAK	%	5	E	U	e	u	Ö	ï	●	µ	≈	’		
6	ACK	SYN	&	6	F	V	f	v	Ü	ñ	¶	∂	∆	÷		
7	BEL	ETB	'	7	G	W	g	w	á	ó	ß	∑	«	◇		
8	BS	CAN	(	8	H	X	h	x	à	ò	®	∏	»	ÿ		
9	HT	EM	)	9	I	Y	i	y	â	ô	©	π	…			
A	LF	SUB	*	:	J	Z	j	z	ä	ö	™	∫	⎵			
B	VT	ESC	+	;	K	[	k	{	ã	õ	´	ª	À			
C	FF	FS	,	<	L	\	l	|	å	ú	¨	º	Ã			
D	CR	GS	-	=	M	]	m	}	ç	ù	≠	Ω	Õ			
E	SO	RS	.	>	N	^	n	~	é	û	Æ	æ	Œ			
F	SI	US	/	?	O	_	o	DEL	è	ü	Ø	ø	œ			

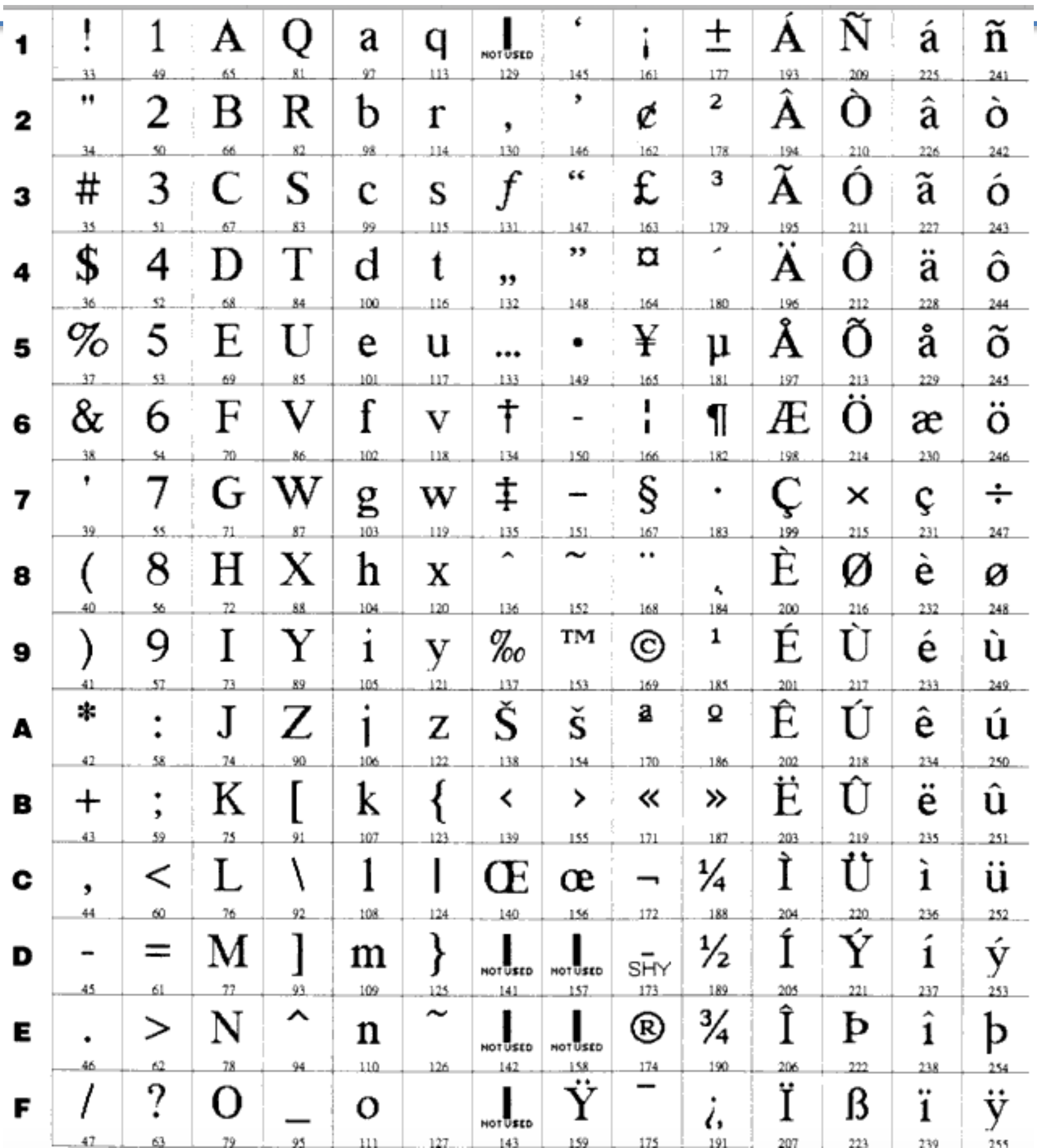
⎵ stands for a nonbreaking space, the same width as a digit.
The shaded characters cannot normally be generated from the Macintosh keyboard or keypad.

Figure 1. Macintosh Character Set

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

MS Windows (1252)

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023



1	! 33	1 49	A 65	Q 81	a 97	q 113	NOT USED 129	‘ 145	¡ 161	± 177	Á 193	Ñ 209	á 225	ñ 241
2	" 34	2 50	B 66	R 82	b 98	r 114	‚ 130	’ 146	¢ 162	^{2} 178	Â 194	Ò 210	â 226	ò 242
3	# 35	3 51	C 67	S 83	c 99	s 115	ƒ 131	“ 147	£ 163	^{3} 179	Ã 195	Ó 211	ã 227	ó 243
4	$ 36	4 52	D 68	T 84	d 100	t 116	„ 132	” 148	¤ 164	´ 180	Ä 196	Ô 212	ä 228	ô 244
5	% 37	5 53	E 69	U 85	e 101	u 117	… 133	• 149	¥ 165	µ 181	Å 197	Õ 213	å 229	õ 245
6	& 38	6 54	F 70	V 86	f 102	v 118	† 134	– 150	¦ 166	¶ 182	Æ 198	Ö 214	æ 230	ö 246
7	' 39	7 55	G 71	W 87	g 103	w 119	‡ 135	— 151	§ 167	· 183	Ç 199	× 215	ç 231	÷ 247
8	(40	8 56	H 72	X 88	h 104	x 120	ˆ 136	˜ 152	¨ 168	¸ 184	È 200	Ø 216	è 232	ø 248
9	) 41	9 57	I 73	Y 89	i 105	y 121	‰ 137	™ 153	© 169	^{1} 185	É 201	Ù 217	é 233	ù 249
A	* 42	: 58	J 74	Z 90	j 106	z 122	Š 138	š 154	ª 170	º 186	Ê 202	Ú 218	ê 234	ú 250
B	+ 43	; 59	K 75	[91	k 107	{ 123	‹ 139	› 155	« 171	» 187	Ë 203	Û 219	ë 235	û 251
C	, 44	< 60	L 76	\ 92	l 108	| 124	Œ 140	œ 156	¬ 172	¼ 188	Ì 204	Ü 220	ì 236	ü 252
D	- 45	= 61	M 77	] 93	m 109	} 125	NOT USED 141	NOT USED 157	SHY 173	½ 189	Í 205	Ý 221	í 237	ý 253
E	. 46	> 62	N 78	^ 94	n 110	~ 126	NOT USED 142	NOT USED 158	® 174	¾ 190	Î 206	Þ 222	î 238	þ 254
F	/ 47	? 63	O 79	_ 95	o 111	127	NOT USED 143	Ÿ 159	¯ 175	¿ 191	Ï 207	ß 223	ï 239	ÿ 255

COMP122

Lab

Software Dev Tools

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

Software Tool Chain

Figure 7.1.6: Assembly language either is written by a programmer or is the output of a compiler (COD Figure A.1.6).

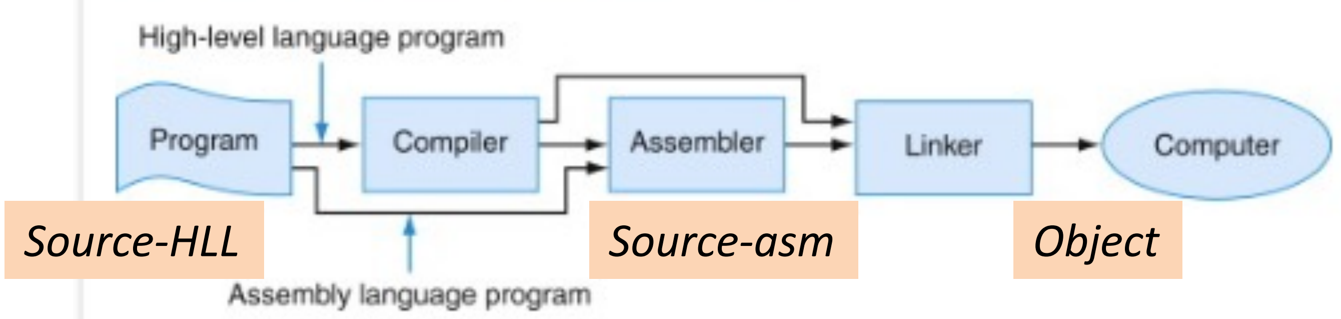
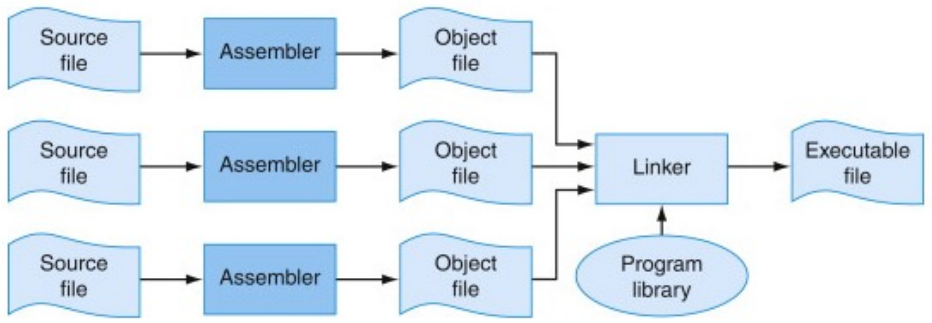


Figure 7.1.1: The process that produces an executable file (COD Figure A.1.1).

An assembler translates a file of assembly language into an object file, which is linked with other files and libraries into an executable file.

Multiple Source files



Software Tool Chain

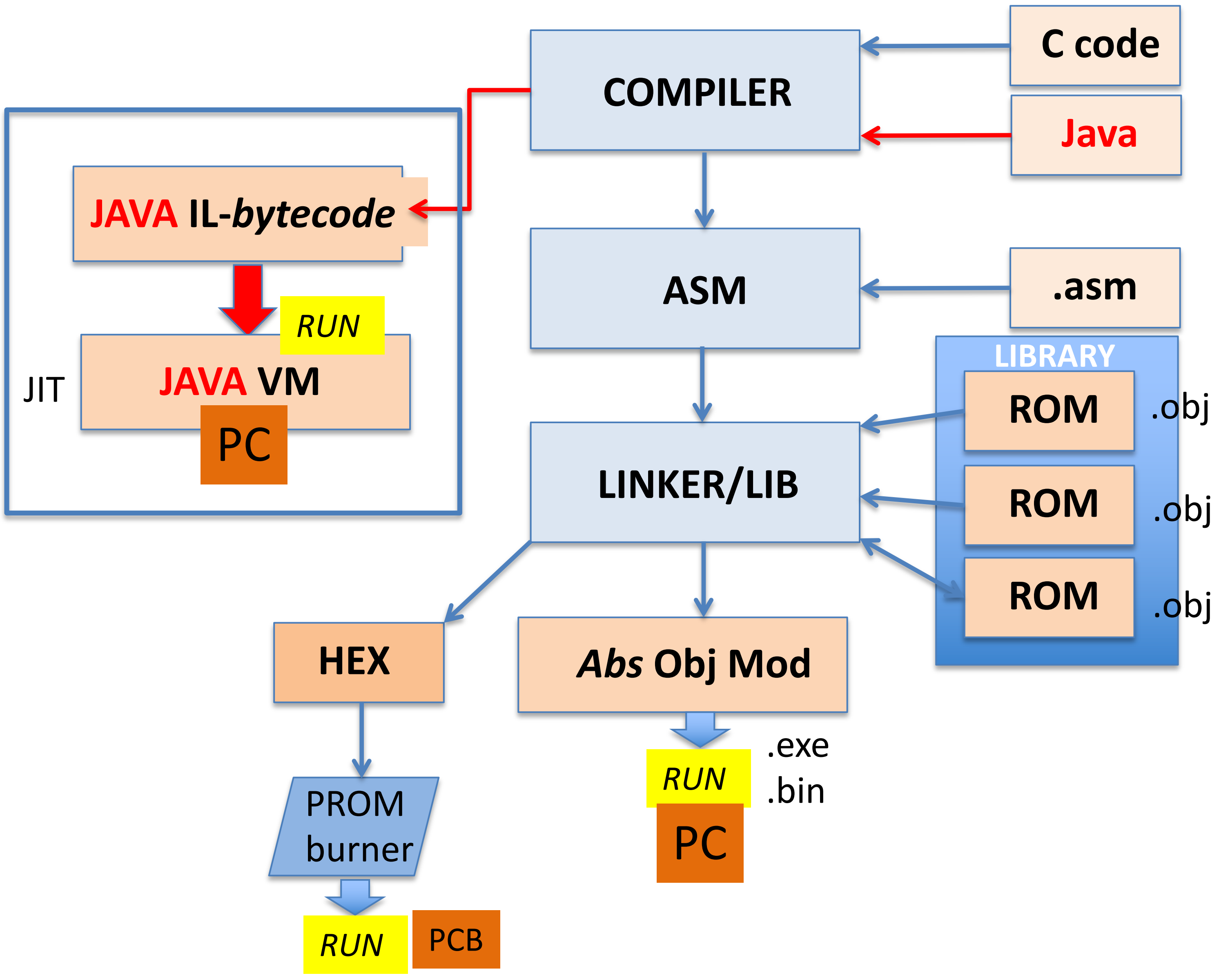
- Compilers
  - *Compiled* languages (C, C++, C#, VB)
    - Compile *completely*: Translate HLL (.c, .h) into ASM (.asm)
  - *Interpreted* languages (**Java**, Pascal)
    - Compile *incompletely* ("JIT") to an "intermediate" language
    - "Pseudo" code is compiled at run time (slow)
- Assemblers
  - Translate ASM (.asm) into linkable machine code modules ("LM")
- Linkers
  - Combine ("link") LM modules into a single "executable" (.exe)
  - Resolve external references
  - Embed calls to dynamic "link libraries" or "frameworks" (.dll files)
- Debuggers

- *SDK contains Compilers + API (Libraries) + IDE*
- *IDE is a development environment w/debugger*

CSUN CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP122

IDE + Platforms



CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

SDK/IDE

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023

SOFTWARE DEV KIT
INTEGRATED DEV ENVIRONMENT

- Compiler
  - **JDK***
  - **Gcc****
- IDE
  - **jGrasp***
  - Eclipse
  - MPlab
- SDK+IDE
  - MIPS *Mars***
  - ARM sim**
  - MS Visual Studio
  - Apple Xcode

*Used for **Java** (COMP110)

Used for ***Assembler* (COMP122)

Development Platforms

Software Applications:
Development Platforms

- Design
  - Microsoft
    - OS = Windows (7, 8, 10)
    - API = .NET Framework
    - SDK/IDE = **Visual Studio** *Cross Platform*
    - Languages = .NET versions of VB, C#, C++, Java
  - Apple
    - OS = Mac OS X, iOS (mobile)
    - API = Xcode (Cocoa Touch)
    - SDK/IDE = **Xcode**
    - Languages = Objective C, Swift
  - Google
    - OS = Android
    - API = Android
    - SDK/IDE = **Android**
    - Languages = C++

SDK/IDE

COMPILER - ASSEMBLER

❖ ***SDK*** = ***IDE*** + Compiler (for C/C++)

- Eclipse
- **Gnu gcc** (Gnu C compiler)
  - Windows
  - Pi (MinGW)
- **MIPS *MARS*** (assembler+simulator)
- **ARM Sim**
- Others (misc)
  - MS Visual Studio
  - IAR

COMP122

Dev Boards

POPULAR DEVELOPMENT BOARDS

SoCKit Development Kit

The SoCKit Development Kit presents a robust hardware design platform built around the Altera System-on-Chip (SoC) FPGA, which combines the latest dual-core Cortex-A9 embedded cores with industry-leading programmable logic for ultimate design flexibility

MAX1000 IOT Maker Board

For engineers designing compact smart solutions for the IoT market, this FPGA IoT Maker Board is an excellent tool to speed up the development process and enter the market with a high-performance, reliable product.

Google Coral Dev Board

The Coral Dev Board is now in-stock and available for free 1-day shipping at Arrow.com. Prototype, scale, and deploy with more flexibility using the Coral Dev Board and accessories with Google.

Software

Platforms

Software Platforms

- ***Standalone*** Applications
  - Native
    - Desktop
    - Mobil *apps* (phone/tablet)
  - Web
    - Client (“Front end” via browser)
    - Server (”Back end”)
- ***Embedded Control***
  - Appliances
  - Cars/airplanes
  - Phones/tablets
    - iOS
    - Android
  - Computer Peripherals
    - Storage devices
    - Printers

App Types

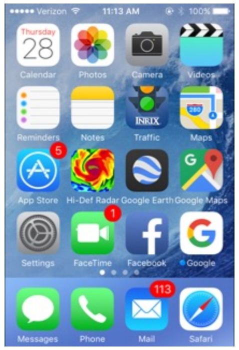
- NATIVE
  - Runs directly on the device/computer on its OS
    - ***Computer*** (desktop or laptop)
    - ***Mobile*** (phone or tablet)

- WEB
  - Runs remotely on the website server and is *displayed* on the device/computer via its ***Browser***
- Mobile Web Apps
  - redesigned websites for display on *mobile devices* (phones, tablets) that include applications ("Web Apps")

COMP122

Standalone Platforms

- ❖ Standalone Applications
  - ❑ Native
    - ➢ Desktop
      - ▪ Universal ("Office")
      - ▪ **Specialized <- THIS CLASS**
    - ➢ Mobil *apps* (phone/tablet)
  - ❑ Web
    - ➢ Client ("Front end" via browser)
      - ▪ Desktop
      - ▪ Mobil
    - ➢ Server ("Back end")



Running/Debugging

RUN

Embedded Systems

SIMULATOR

SOFTWARE PC

- ❑ Debugger runtime environment in (IDE)
- ❑ Breakpoints
- ❑ Watch variables
- ❑ Target device selection
  - PC
  - Phone/tablet
  - *Board* →

- MIPS *Mars*
- ARM Sim

EMULATOR

Substitute HARDWARE PCB

- ❑ **ICE** (in-circuit)
  - Pods
  - Breakpoints
  - Trace triggers & buffers
- ❑ Memory (known good)
  - R/W (ROM/RAM)
  - Wait states

HARDWARE

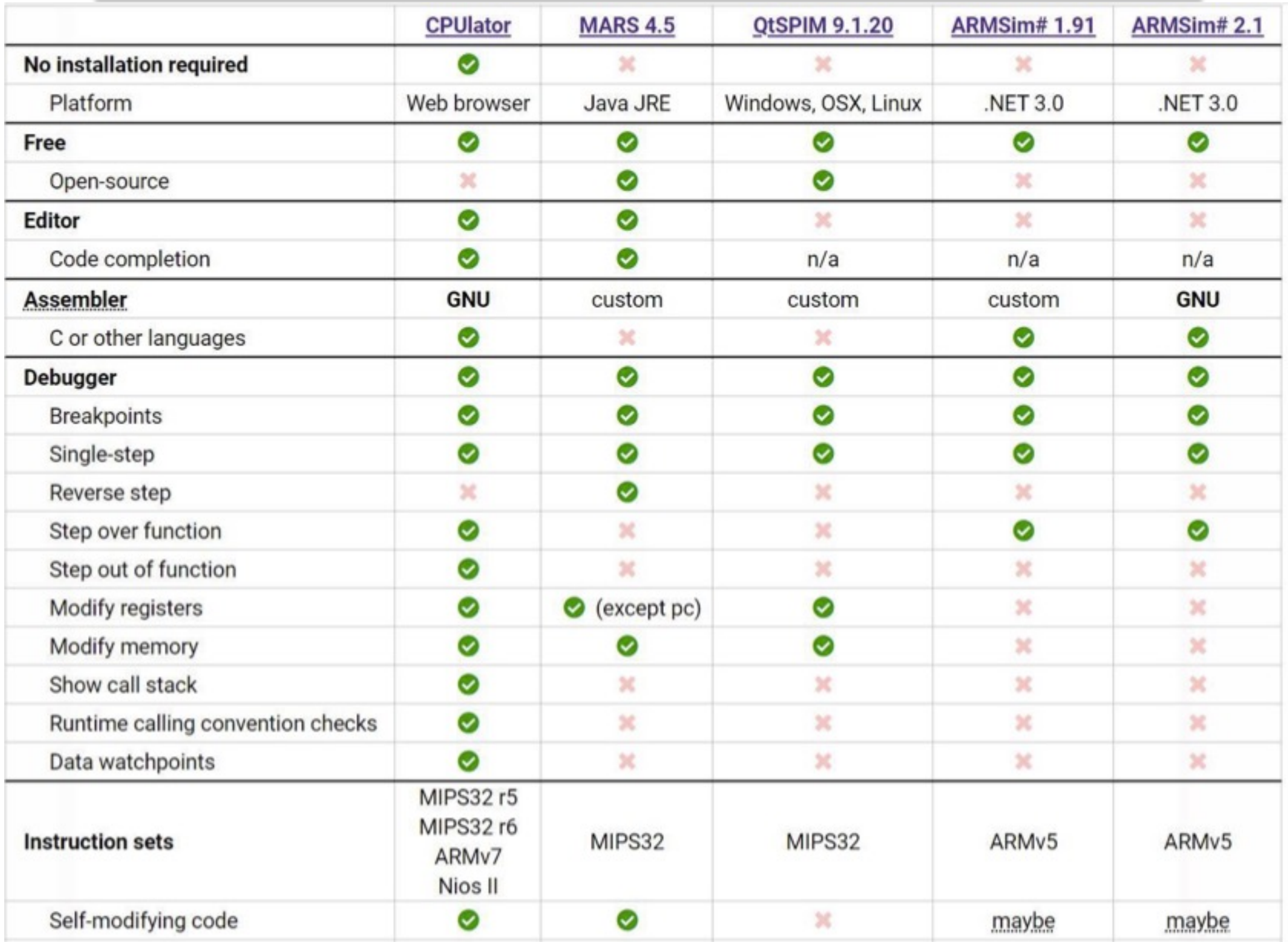
Actual HARDWARE PCB

- ❑ Code burned into ROM
- ❑ Working RAM
- ❑ Can use ICE
- ❑ Board bring-up
- ❑ Built-in test
  - JTAG
- ❑ Logic analyzers

Raspberry Pi

COMP122

Simulators

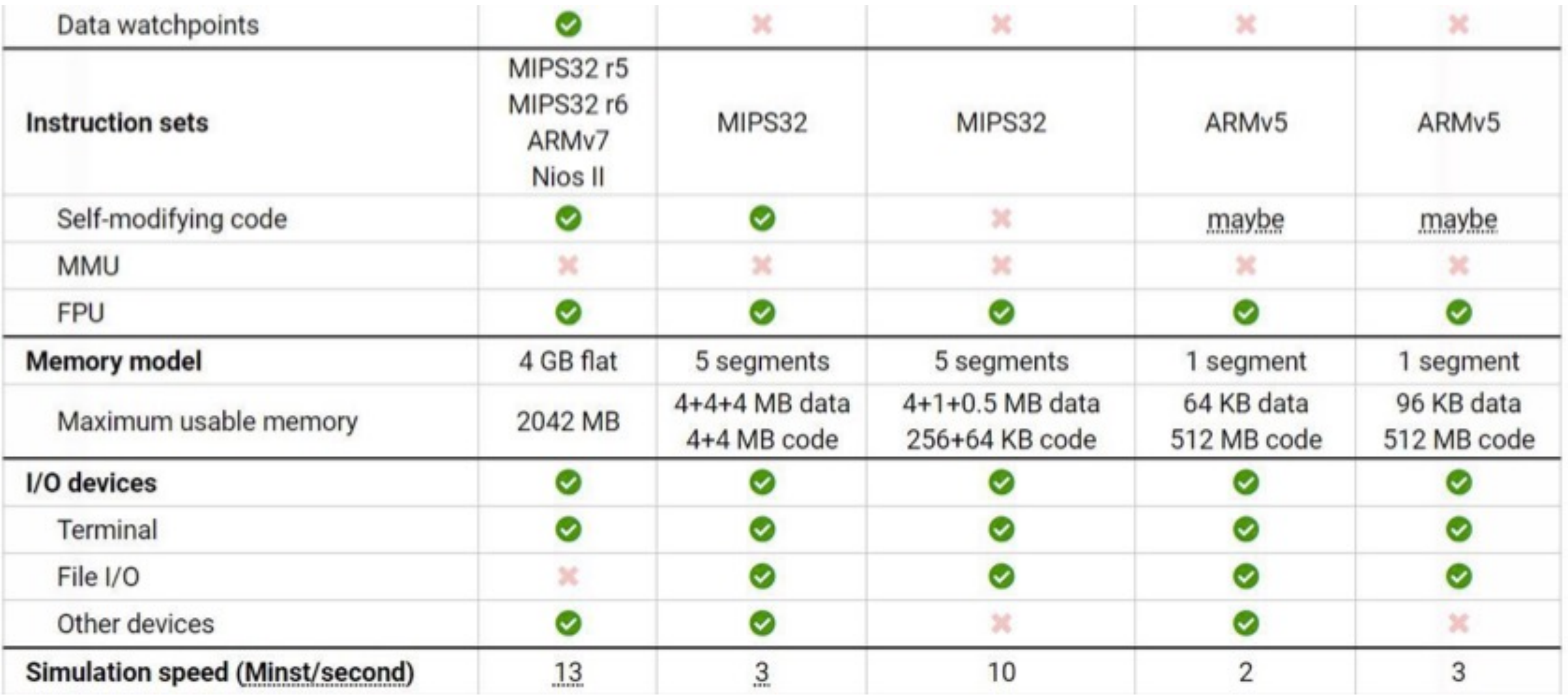


	CPUlator	MARS 4.5	QtSPIM 9.1.20	ARMSim# 1.91	ARMSim# 2.1
No installation required	✓	✗	✗	✗	✗
Platform	Web browser	Java JRE	Windows, OSX, Linux	.NET 3.0	.NET 3.0
Free	✓	✓	✓	✓	✓
Open-source	✗	✓	✓	✗	✗
Editor	✓	✓	✗	✗	✗
Code completion	✓	✓	n/a	n/a	n/a
Assembler	**GNU**	custom	custom	custom	**GNU**
C or other languages	✓	✗	✗	✓	✓
Debugger	✓	✓	✓	✓	✓
Breakpoints	✓	✓	✓	✓	✓
Single-step	✓	✓	✓	✓	✓
Reverse step	✗	✓	✗	✗	✗
Step over function	✓	✗	✗	✓	✓
Step out of function	✓	✗	✗	✗	✗
Modify registers	✓	✓ (except pc)	✓	✗	✗
Modify memory	✓	✓	✓	✗	✗
Show call stack	✓	✗	✗	✗	✗
Runtime calling convention checks	✓	✗	✗	✗	✗
Data watchpoints	✓	✗	✗	✗	✗
Instruction sets	MIPS32 r5 MIPS32 r6 ARMv7 Nios II	MIPS32	MIPS32	ARMv5	ARMv5
Self-modifying code	✓	✓	✗	maybe	maybe

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

Simulators

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman 2016-2023



Data watchpoints	✓	✗	✗	✗	✗
Instruction sets	MIPS32 r5 MIPS32 r6 ARMv7 Nios II	MIPS32	MIPS32	ARMv5	ARMv5
Self-modifying code	✓	✓	✗	maybe	maybe
MMU	✗	✗	✗	✗	✗
FPU	✓	✓	✓	✓	✓
Memory model	4 GB flat	5 segments	5 segments	1 segment	1 segment
Maximum usable memory	2042 MB	4+4+4 MB data 4+4 MB code	4+1+0.5 MB data 256+64 KB code	64 KB data 512 MB code	96 KB data 512 MB code
I/O devices	✓	✓	✓	✓	✓
Terminal	✓	✓	✓	✓	✓
File I/O	✗	✓	✓	✓	✓
Other devices	✓	✓	✗	✓	✗
Simulation speed (Minst/second)	13	3	10	2	3

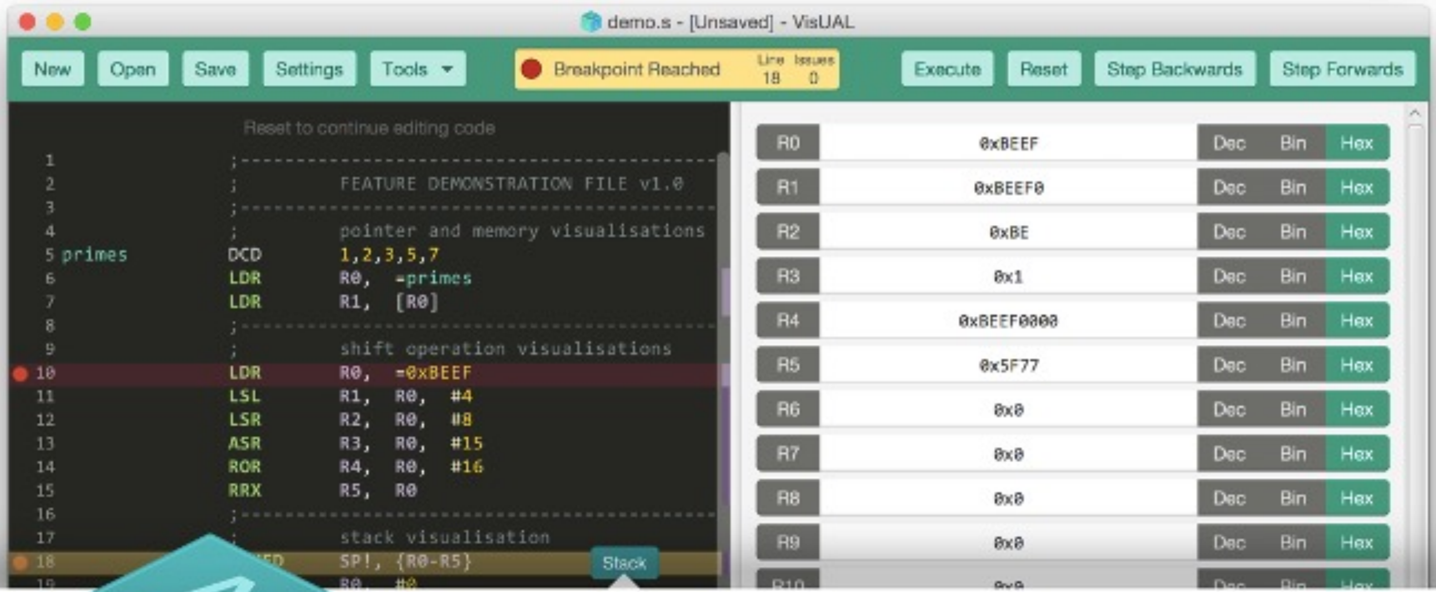
CSUN CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP122

Simulators

DR JEFF SOFTWARE INDIE APP DEVELOPER

© Jeff Drobman 2016-2023



```
Reset to continue editing code
        ;------------------------------------------
        ;         FEATURE DEMONSTRATION FILE v1.0
        ;------------------------------------------
        ;         pointer and memory visualisations
primes  DCD       1,2,3,5,7
        LDR       R0,  =primes
        LDR       R1,  [R0]
        ;------------------------------------------
        ;         shift operation visualisations
        LDR       R0,  =0xBEEF
        LSL       R1,  R0,  #4
        LSR       R2,  R0,  #8
        ASR       R3,  R0,  #15
        ROR       R4,  R0,  #16
        RRX       R5,  R0
        ;------------------------------------------
        ;         stack visualisation
                  SP!, {R0-R5}
                  R0,  #0
```

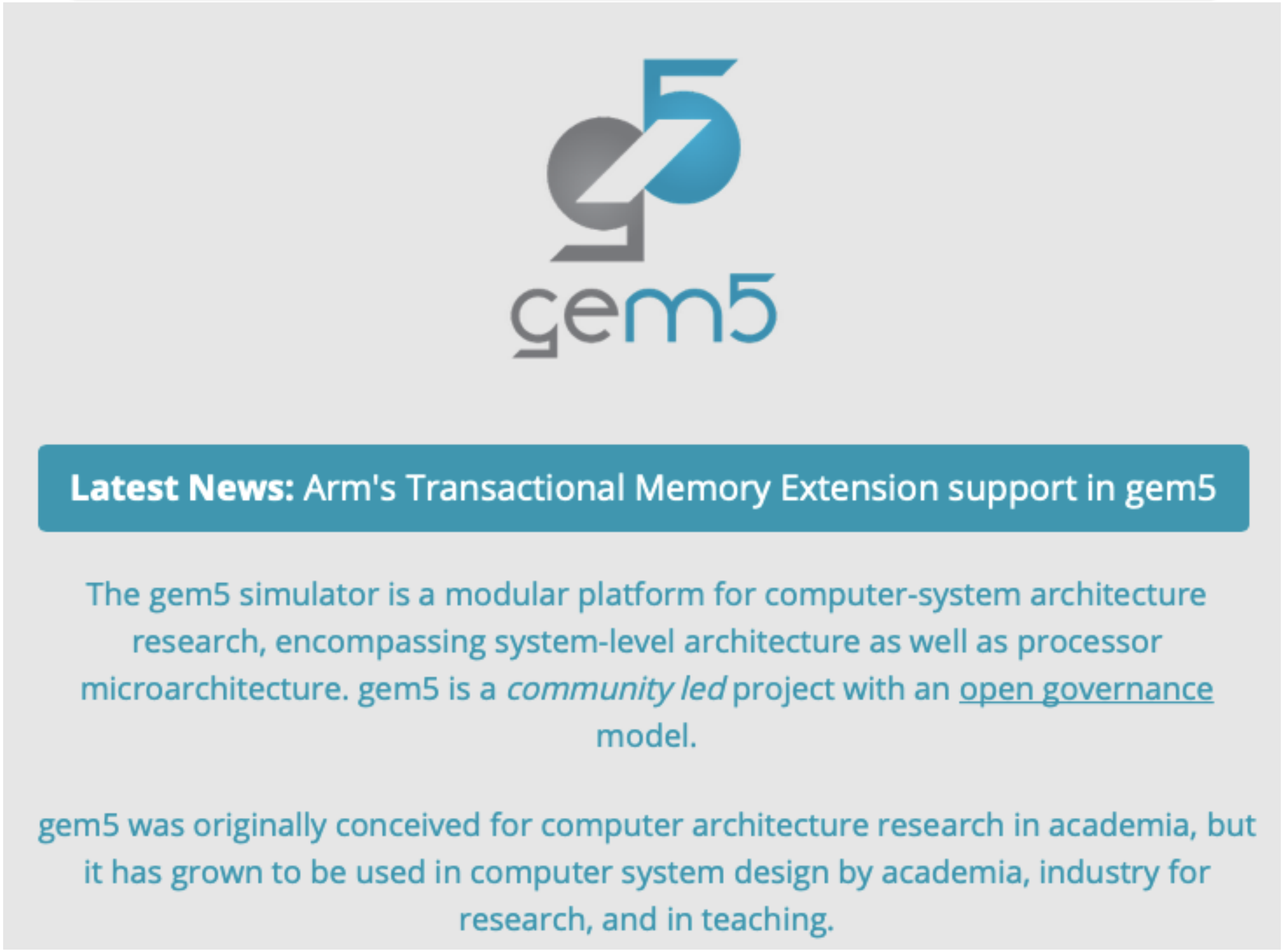
```
R0    0xBEEF        Dec  Bin  Hex
R1    0xBEEF0       Dec  Bin  Hex
R2    0xBE          Dec  Bin  Hex
R3    0x1           Dec  Bin  Hex
R4    0xBEEF0000    Dec  Bin  Hex
R5    0x5F77        Dec  Bin  Hex
R6    0x0           Dec  Bin  Hex
R7    0x0           Dec  Bin  Hex
R8    0x0           Dec  Bin  Hex
R9    0x0           Dec  Bin  Hex
R10   0x0           Dec  Bin  Hex
```

VisUAL

A highly visual ARM emulator

Visual is a highly visual Arm Emulator that makes programming in ARM assembly more accessible. It was created almost specifically to help Computer science student get through the rigorous Introduction to Computer architecture course. Visual strongest positive is perhaps its

Simulators



Latest News: Arm's Transactional Memory Extension support in gem5

The gem5 simulator is a modular platform for computer-system architecture research, encompassing system-level architecture as well as processor microarchitecture. gem5 is a *community led* project with an open governance model.

gem5 was originally conceived for computer architecture research in academia, but it has grown to be used in computer system design by academia, industry for research, and in teaching.

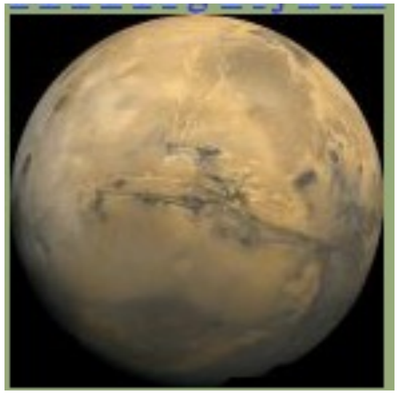
CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

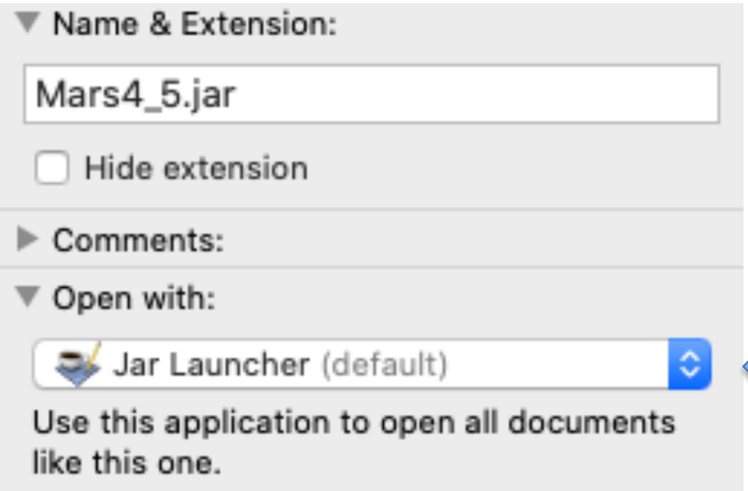
Lab

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023

MIPS

MARS





Name & Extension:

Mars4_5.jar

☐ Hide extension

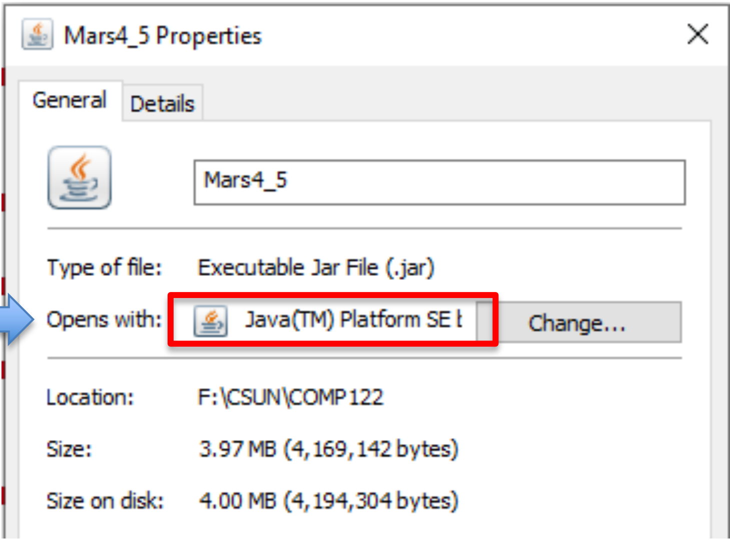
Comments:

Open with:

Jar Launcher (default)

Use this application to open all documents like this one.

Windows 10



Mars4_5 Properties

General | Details

Mars4_5

Type of file:	Executable Jar File (.jar)
Opens with:	Java(TM) Platform SE b
Location:	F:\CSUN\COMP122
Size:	3.97 MB (4,169,142 bytes)
Size on disk:	4.00 MB (4,194,304 bytes)

Change...

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

MARS

MARS ***(MIPS Assembler and Runtime Simulator)***

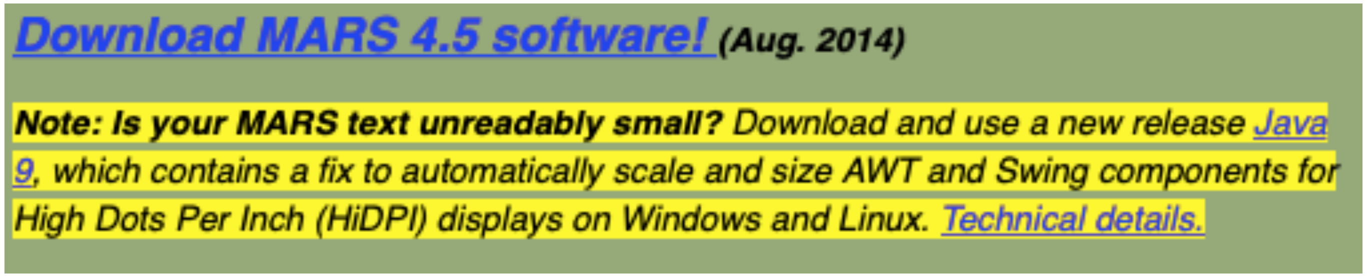
Missouri State
UNIVERSITY

courses.missouristate.edu

Mars4_5.jar

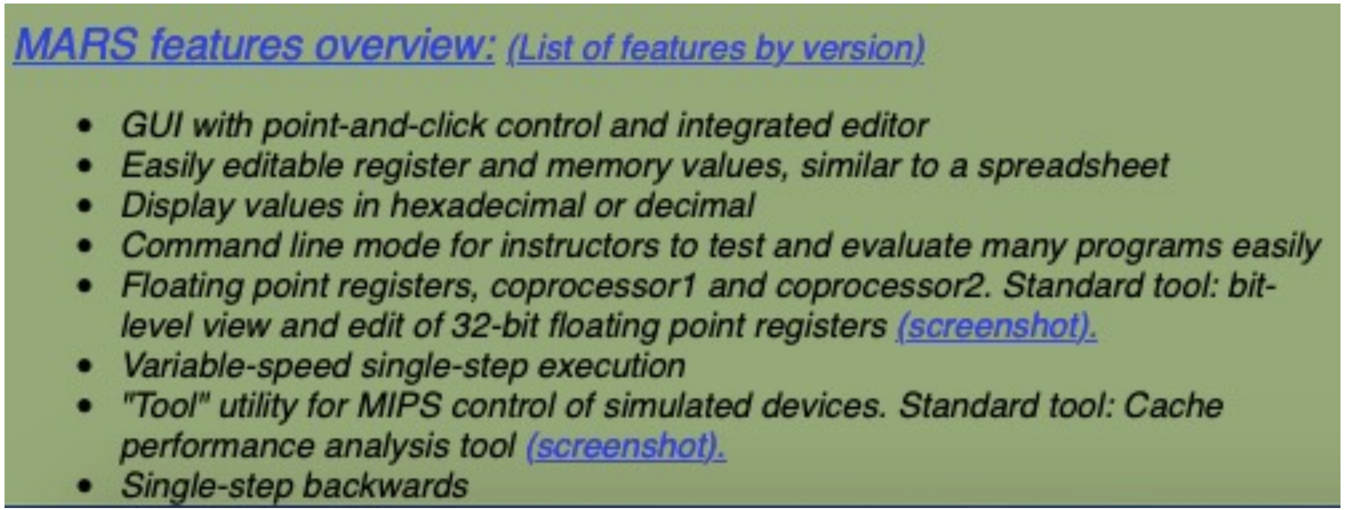
Mac Desktop

https://courses.missouristate.edu/KenVollmar/MARS/download.htm



Download MARS 4.5 software! ***(Aug. 2014)***

Note: Is your MARS text unreadably small? *Download and use a new release Java 9, which contains a fix to automatically scale and size AWT and Swing components for High Dots Per Inch (HiDPI) displays on Windows and Linux. Technical details.*



MARS features overview: (List of features by version)

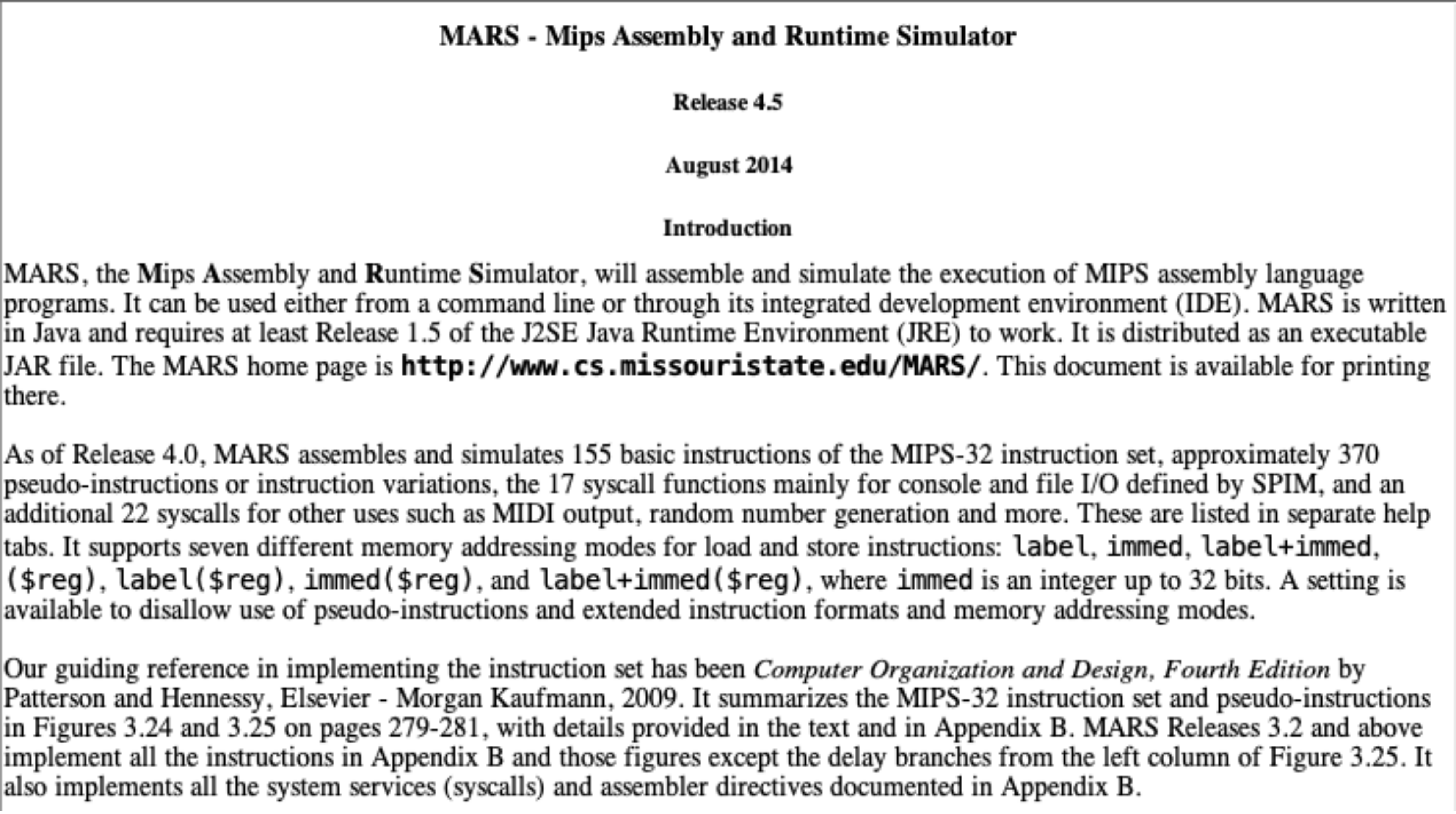
- *GUI with point-and-click control and integrated editor*
- *Easily editable register and memory values, similar to a spreadsheet*
- *Display values in hexadecimal or decimal*
- *Command line mode for instructors to test and evaluate many programs easily*
- *Floating point registers, coprocessor1 and coprocessor2. Standard tool: bit-level view and edit of 32-bit floating point registers (screenshot).*
- *Variable-speed single-step execution*
- *"Tool" utility for MIPS control of simulated devices. Standard tool: Cache performance analysis tool (screenshot).*
- *Single-step backwards*

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

DR JEFF SOFTWARE
INDIE APP DEVELOPER

MARS

MARS (MIPS Assembler and Runtime Simulator)



MARS - Mips Assembly and Runtime Simulator

Release 4.5

August 2014

Introduction

MARS, the **M**ips **A**ssembly and **R**untime **S**imulator, will assemble and simulate the execution of MIPS assembly language programs. It can be used either from a command line or through its integrated development environment (IDE). MARS is written in Java and requires at least Release 1.5 of the J2SE Java Runtime Environment (JRE) to work. It is distributed as an executable JAR file. The MARS home page is **`http://www.cs.missouristate.edu/MARS/`**. This document is available for printing there.

As of Release 4.0, MARS assembles and simulates 155 basic instructions of the MIPS-32 instruction set, approximately 370 pseudo-instructions or instruction variations, the 17 syscall functions mainly for console and file I/O defined by SPIM, and an additional 22 syscalls for other uses such as MIDI output, random number generation and more. These are listed in separate help tabs. It supports seven different memory addressing modes for load and store instructions: `label`, `immed`, `label+immed`, `($reg)`, `label($reg)`, `immed($reg)`, and `label+immed($reg)`, where `immed` is an integer up to 32 bits. A setting is available to disallow use of pseudo-instructions and extended instruction formats and memory addressing modes.

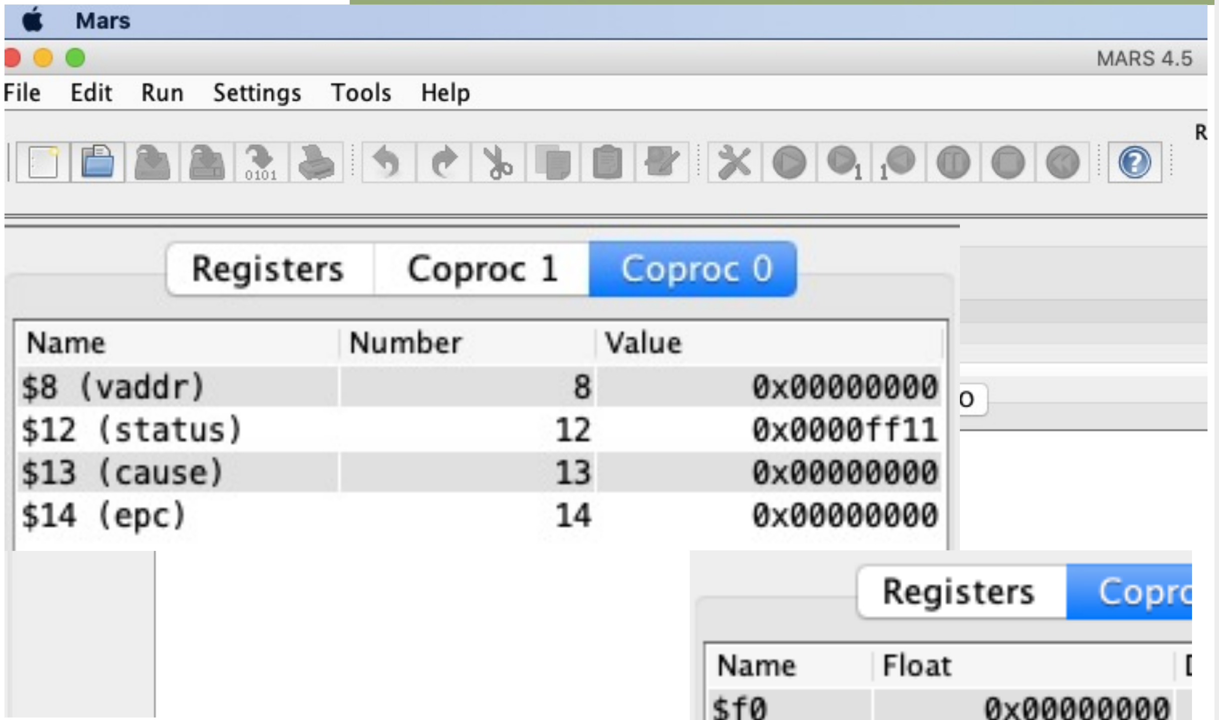
Our guiding reference in implementing the instruction set has been *Computer Organization and Design, Fourth Edition* by Patterson and Hennessy, Elsevier - Morgan Kaufmann, 2009. It summarizes the MIPS-32 instruction set and pseudo-instructions in Figures 3.24 and 3.25 on pages 279-281, with details provided in the text and in Appendix B. MARS Releases 3.2 and above implement all the instructions in Appendix B and those figures except the delay branches from the left column of Figure 3.25. It also implements all the system services (syscalls) and assembler directives documented in Appendix B.

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

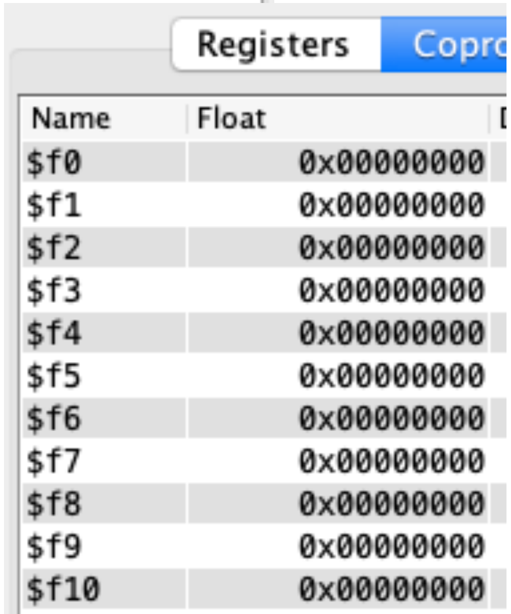
MARS

MARS (MIPS Assembler and Runtime Simulator)

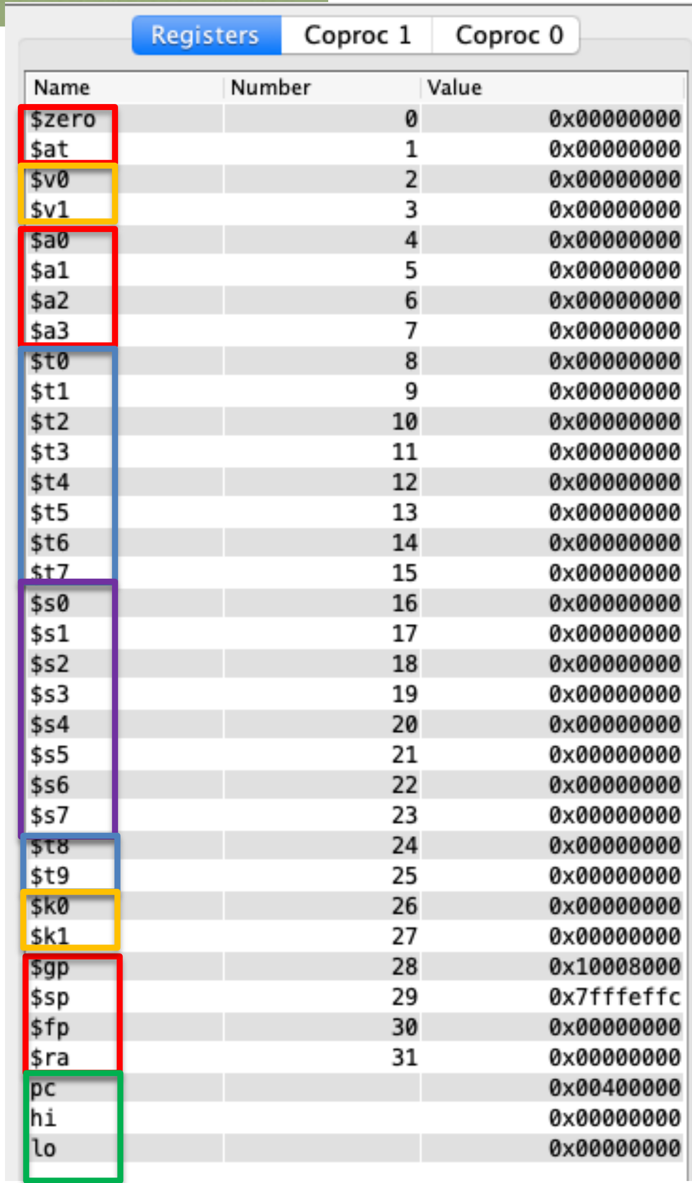
Registers



Name	Number	Value
$8 (vaddr)	8	0x00000000
$12 (status)	12	0x0000ff11
$13 (cause)	13	0x00000000
$14 (epc)	14	0x00000000



Name	Float
$f0	0x00000000
$f1	0x00000000
$f2	0x00000000
$f3	0x00000000
$f4	0x00000000
$f5	0x00000000
$f6	0x00000000
$f7	0x00000000
$f8	0x00000000
$f9	0x00000000
$f10	0x00000000

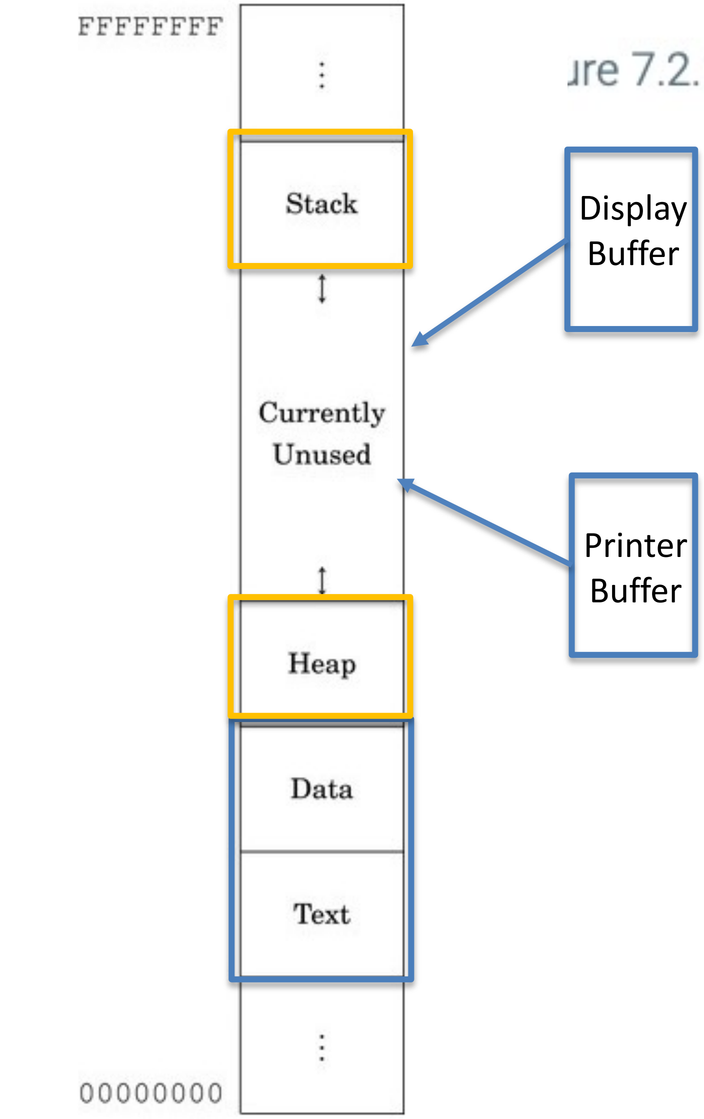


Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

Memory Segments

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023



Typical memory layout for a program with a 32-bit address space.

ıre 7.2.1: Object file (COD Figure A.2.1).

A UNIX assembler produces an object file with six distinct sections.

Object file header	Text segment	Data segment	Relocation information	Symbol table	Debugging information

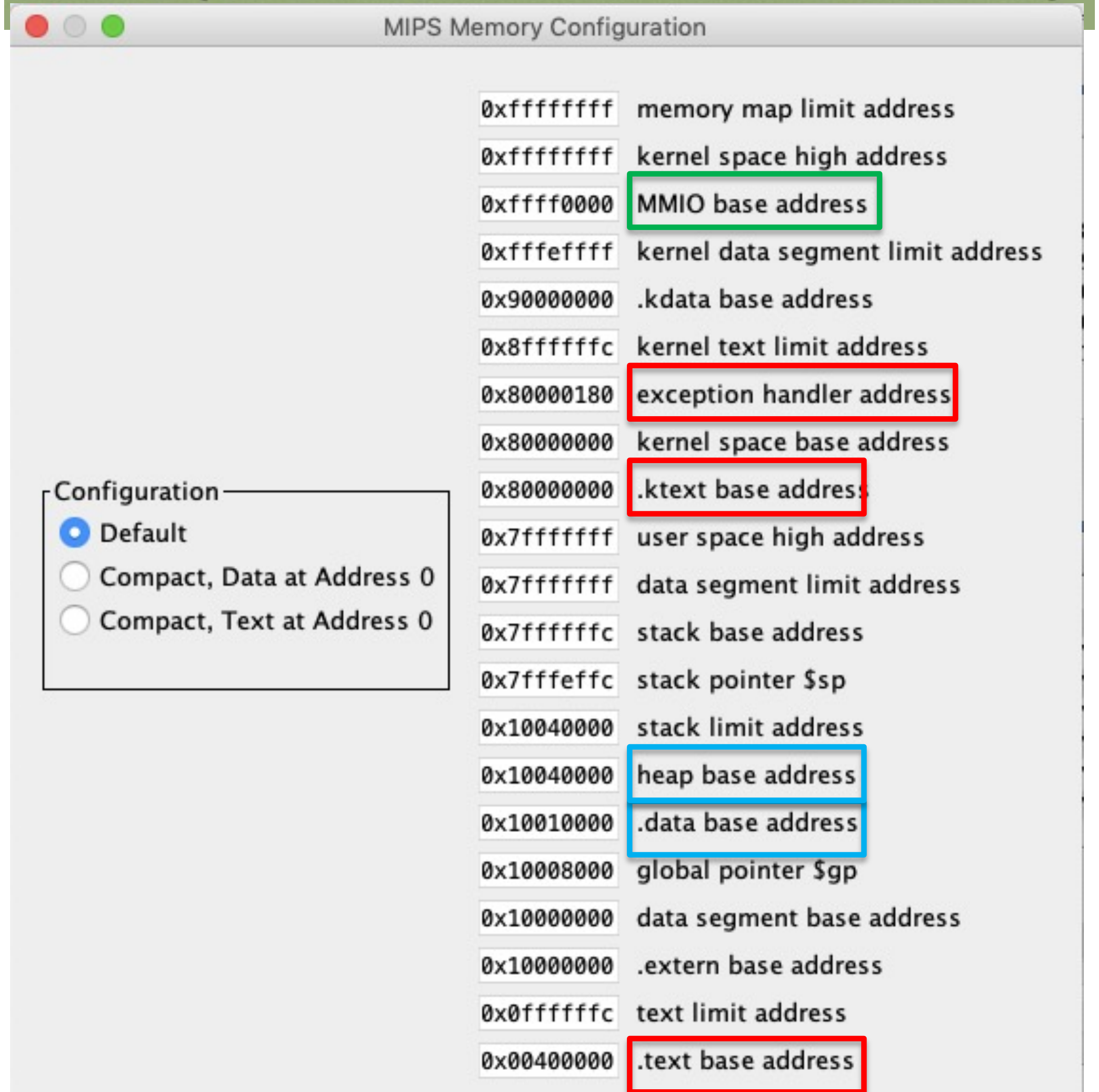
CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

MARS

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman 2016-2023

MARS (MIPS Assembler and Runtime Simulator)

Memory Map


MIPS Memory Configuration

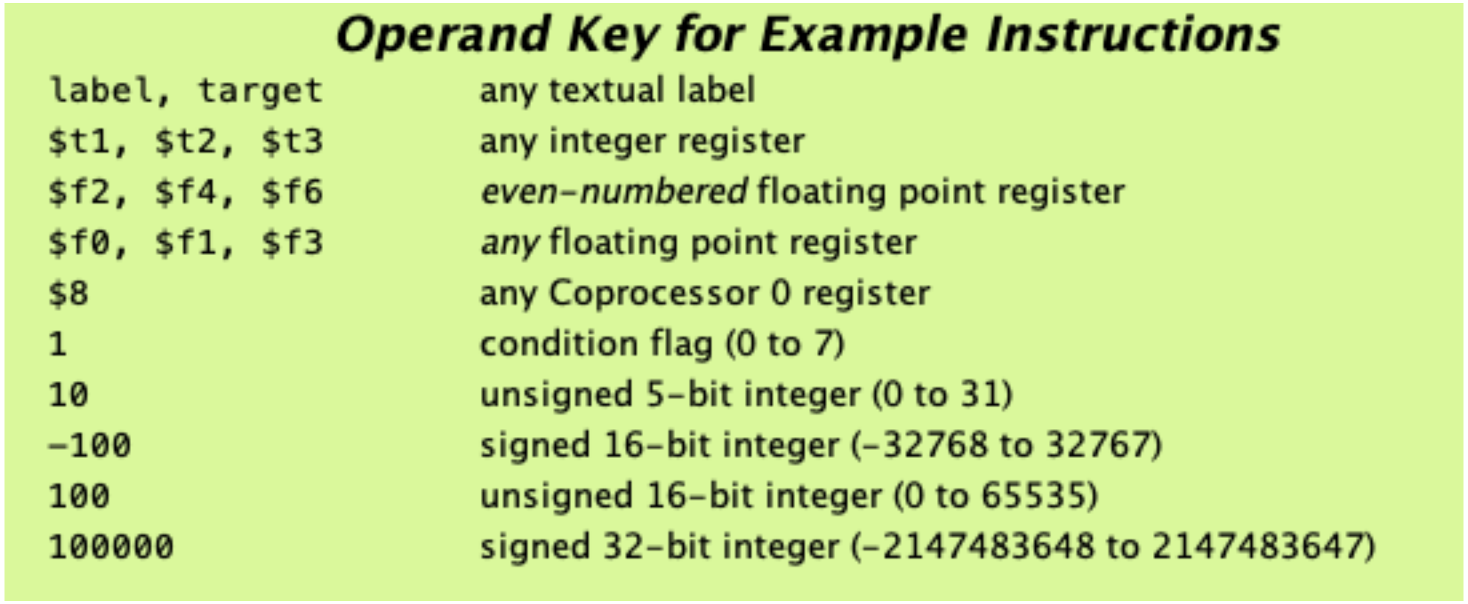
Configuration
[✓ Default]
Compact, Data at Address 0
Compact, Text at Address 0

0xffffffff	memory map limit address
0xffffffff	kernel space high address
0xffff0000	MMIO base address
0xfffeffff	kernel data segment limit address
0x90000000	.kdata base address
0x8ffffffc	kernel text limit address
0x80000180	exception handler address
0x80000000	kernel space base address
0x80000000	.ktext base address
0x7fffffff	user space high address
0x7fffffff	data segment limit address
0x7ffffffc	stack base address
0x7fffeffc	stack pointer $sp
0x10040000	stack limit address
0x10040000	heap base address
0x10010000	.data base address
0x10008000	global pointer $gp
0x10000000	data segment base address
0x10000000	.extern base address
0x0ffffffc	text limit address
0x00400000	.text base address

MARS

MARS ***(MIPS Assembler and Runtime Simulator)***

Key



Operand Key for Example Instructions

`label, target`	any textual label
`$t1, $t2, $t3`	any integer register
`$f2, $f4, $f6`	*even-numbered* floating point register
`$f0, $f1, $f3`	*any* floating point register
`$8`	any Coprocessor 0 register
`1`	condition flag (0 to 7)
`10`	unsigned 5-bit integer (0 to 31)
`-100`	signed 16-bit integer (-32768 to 32767)
`100`	unsigned 16-bit integer (0 to 65535)
`100000`	signed 32-bit integer (-2147483648 to 2147483647)

CSUN CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP122

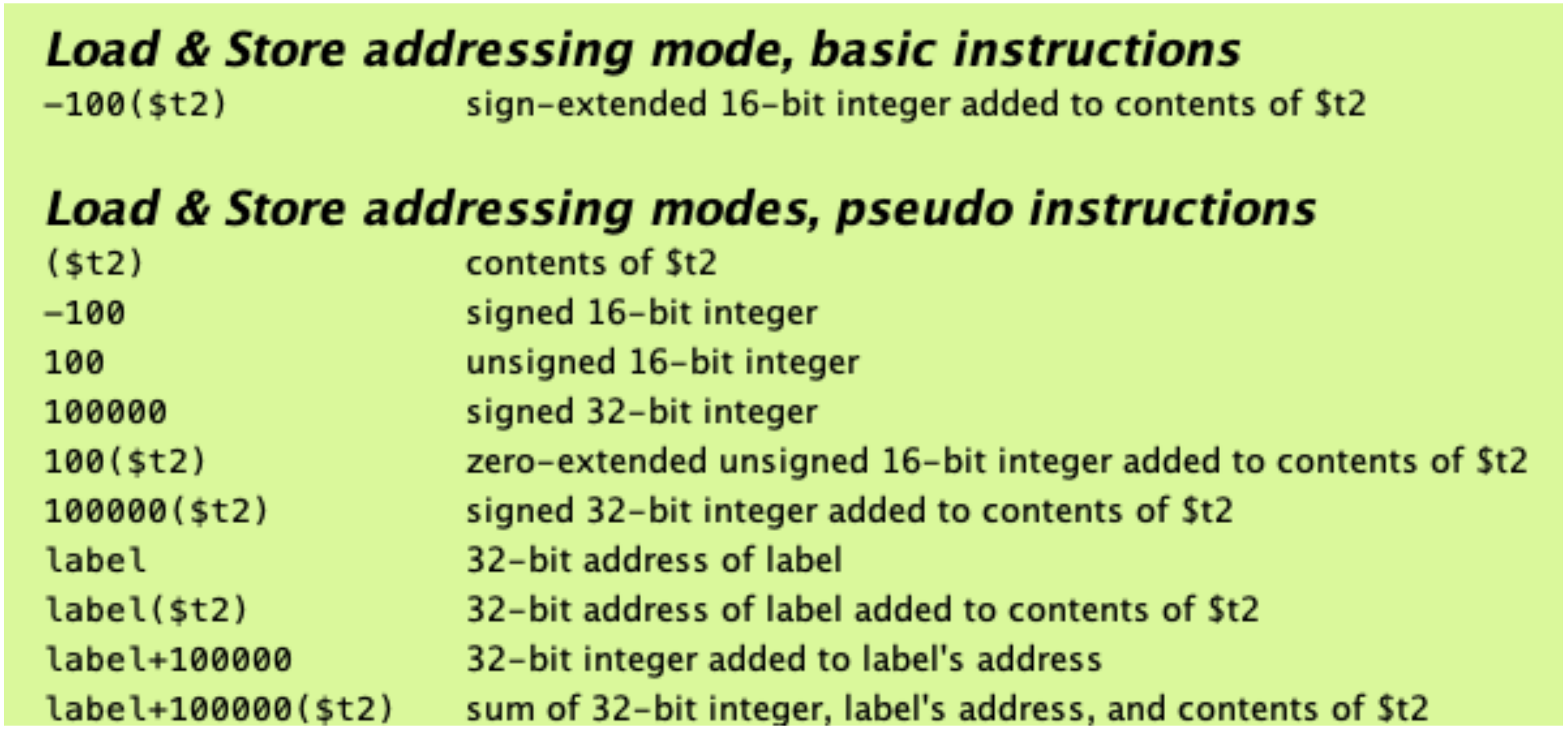
MARS

DR JEFF SOFTWARE INDIE APP DEVELOPER

© Jeff Drobman 2016-2023

MARS ***(MIPS Assembler and Runtime Simulator)***

Addr Modes



Load & Store addressing mode, basic instructions

`-100($t2)`	sign-extended 16-bit integer added to contents of $t2

Load & Store addressing modes, pseudo instructions

`($t2)`	contents of $t2
`-100`	signed 16-bit integer
`100`	unsigned 16-bit integer
`100000`	signed 32-bit integer
`100($t2)`	zero-extended unsigned 16-bit integer added to contents of $t2
`100000($t2)`	signed 32-bit integer added to contents of $t2
`label`	32-bit address of label
`label($t2)`	32-bit address of label added to contents of $t2
`label+100000`	32-bit integer added to label's address
`label+100000($t2)`	sum of 32-bit integer, label's address, and contents of $t2

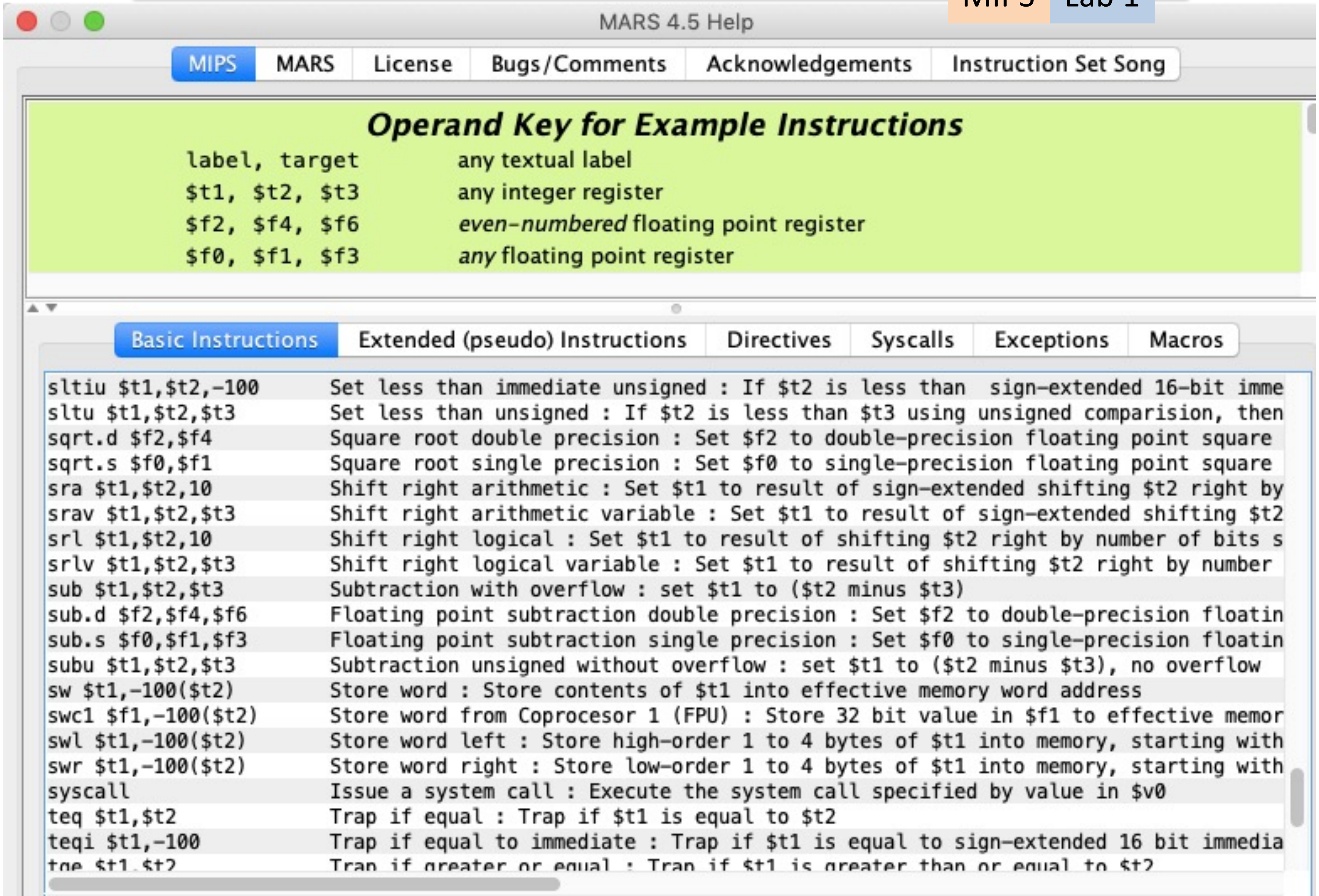
CSUN CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP122

MIPS Assembly

DR JEFF SOFTWARE INDIE APP DEVELOPER © Jeff Drobman 2016-2023

MIPS Lab 1



Operand Key for Example Instructions

`label, target`	any textual label
`$t1, $t2, $t3`	any integer register
`$f2, $f4, $f6`	*even-numbered* floating point register
`$f0, $f1, $f3`	*any* floating point register

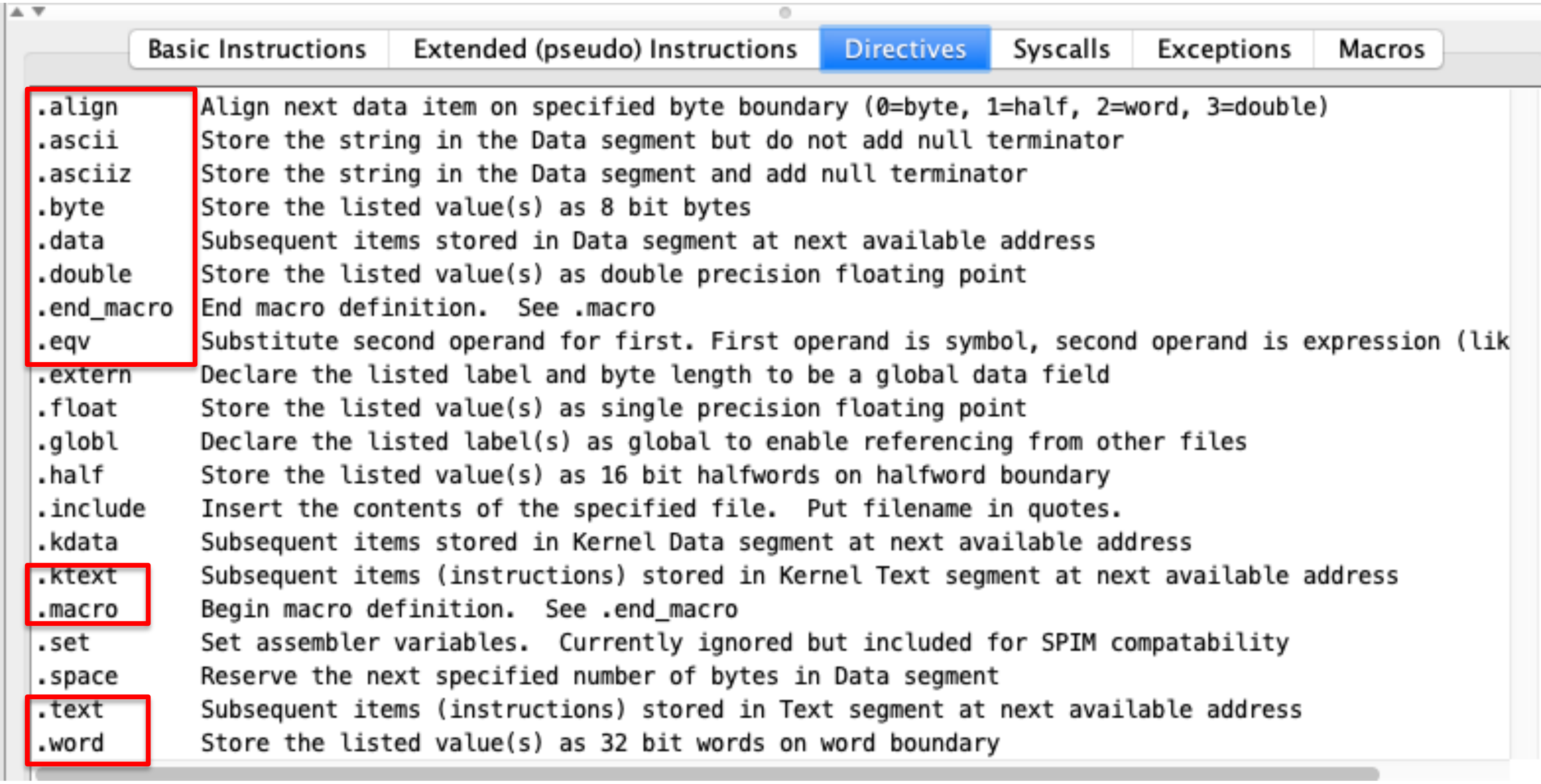
```
sltiu $t1,$t2,-100       Set less than immediate unsigned : If $t2 is less than  sign-extended 16-bit imme
sltu $t1,$t2,$t3         Set less than unsigned : If $t2 is less than $t3 using unsigned comparision, then
sqrt.d $f2,$f4           Square root double precision : Set $f2 to double-precision floating point square
sqrt.s $f0,$f1           Square root single precision : Set $f0 to single-precision floating point square
sra $t1,$t2,10           Shift right arithmetic : Set $t1 to result of sign-extended shifting $t2 right by
srav $t1,$t2,$t3         Shift right arithmetic variable : Set $t1 to result of sign-extended shifting $t2
srl $t1,$t2,10           Shift right logical : Set $t1 to result of shifting $t2 right by number of bits s
srlv $t1,$t2,$t3         Shift right logical variable : Set $t1 to result of shifting $t2 right by number
sub $t1,$t2,$t3          Subtraction with overflow : set $t1 to ($t2 minus $t3)
sub.d $f2,$f4,$f6        Floating point subtraction double precision : Set $f2 to double-precision floatin
sub.s $f0,$f1,$f3        Floating point subtraction single precision : Set $f0 to single-precision floatin
subu $t1,$t2,$t3         Subtraction unsigned without overflow : set $t1 to ($t2 minus $t3), no overflow
sw $t1,-100($t2)         Store word : Store contents of $t1 into effective memory word address
swc1 $f1,-100($t2)       Store word from Coprocesor 1 (FPU) : Store 32 bit value in $f1 to effective memor
swl $t1,-100($t2)        Store word left : Store high-order 1 to 4 bytes of $t1 into memory, starting with
swr $t1,-100($t2)        Store word right : Store low-order 1 to 4 bytes of $t1 into memory, starting with
syscall                  Issue a system call : Execute the system call specified by value in $v0
teq $t1,$t2              Trap if equal : Trap if $t1 is equal to $t2
teqi $t1,-100            Trap if equal to immediate : Trap if $t1 is equal to sign-extended 16 bit immedia
tge $t1,$t2              Trap if greater or equal : Trap if $t1 is greater than or equal to $t2
```

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

MIPS Assembly

DR JEFF SOFTWARE
INDIE APP DEVELOPER

MIPS Lab 1



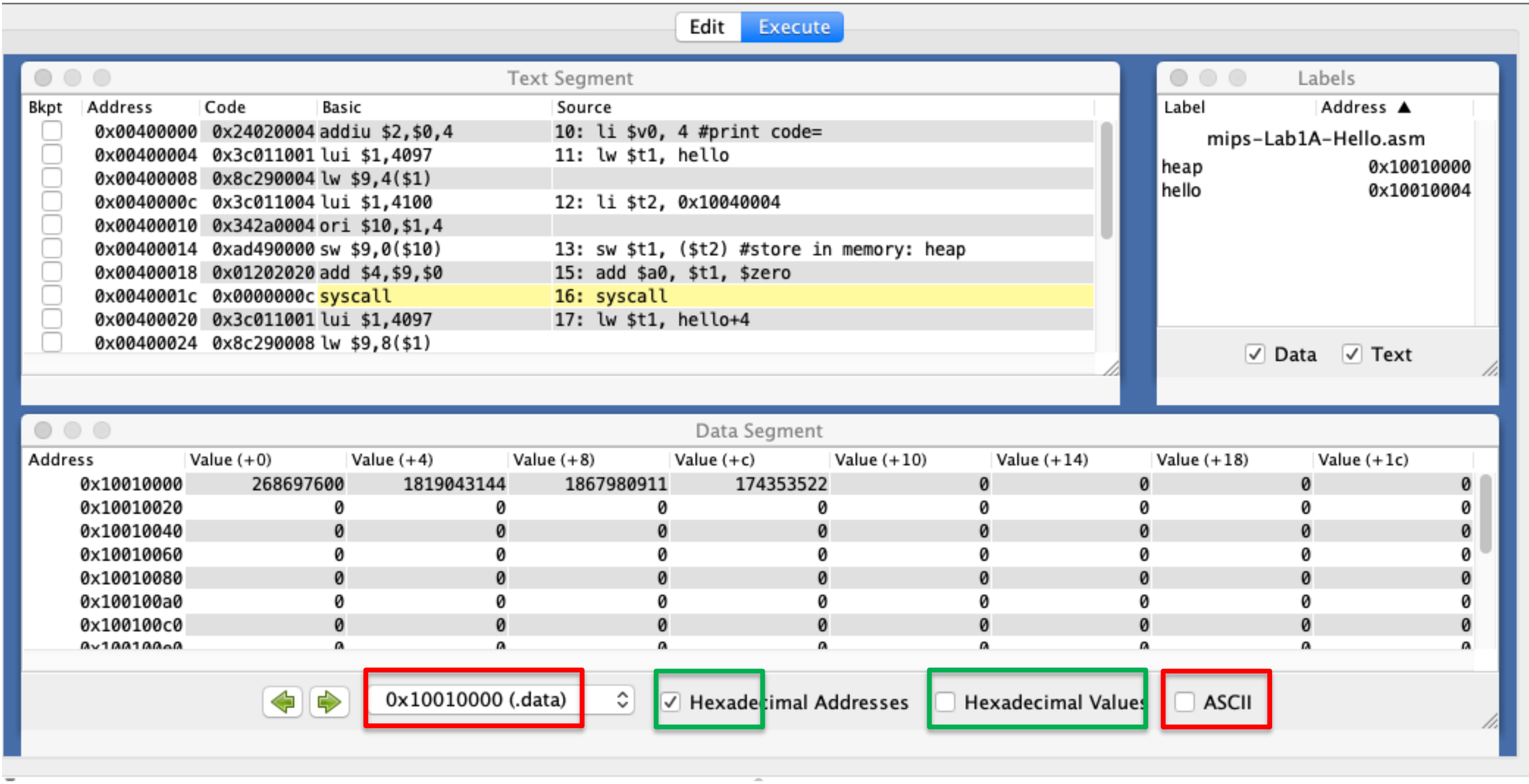
.align	Align next data item on specified byte boundary (0=byte, 1=half, 2=word, 3=double)
.ascii	Store the string in the Data segment but do not add null terminator
.asciiz	Store the string in the Data segment and add null terminator
.byte	Store the listed value(s) as 8 bit bytes
.data	Subsequent items stored in Data segment at next available address
.double	Store the listed value(s) as double precision floating point
.end_macro	End macro definition. See .macro
.eqv	Substitute second operand for first. First operand is symbol, second operand is expression (lik
.extern	Declare the listed label and byte length to be a global data field
.float	Store the listed value(s) as single precision floating point
.globl	Declare the listed label(s) as global to enable referencing from other files
.half	Store the listed value(s) as 16 bit halfwords on halfword boundary
.include	Insert the contents of the specified file. Put filename in quotes.
.kdata	Subsequent items stored in Kernel Data segment at next available address
.ktext	Subsequent items (instructions) stored in Kernel Text segment at next available address
.macro	Begin macro definition. See .end_macro
.set	Set assembler variables. Currently ignored but included for SPIM compatability
.space	Reserve the next specified number of bytes in Data segment
.text	Subsequent items (instructions) stored in Text segment at next available address
.word	Store the listed value(s) as 32 bit words on word boundary

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

MIPS Assembly

DR JEFF SOFTWARE
INDIE APP DEVELOPER

MIPS Lab 1



```
Bkpt  Address     Code        Basic              Source
      0x00400000  0x24020004  addiu $2,$0,4      10:  li $v0, 4 #print code=
      0x00400004  0x3c011001  lui $1,4097        11:  lw $t1, hello
      0x00400008  0x8c290004  lw $9,4($1)
      0x0040000c  0x3c011004  lui $1,4100        12:  li $t2, 0x10040004
      0x00400010  0x342a0004  ori $10,$1,4
      0x00400014  0xad490000  sw $9,0($10)       13:  sw $t1, ($t2) #store in memory: heap
      0x00400018  0x01202020  add $4,$9,$0       15:  add $a0, $t1, $zero
      0x0040001c  0x0000000c  syscall            16:  syscall
      0x00400020  0x3c011001  lui $1,4097        17:  lw $t1, hello+4
      0x00400024  0x8c290008  lw $9,8($1)
```

Label	Address ▲
mips-Lab1A-Hello.asm	
heap	0x10010000
hello	0x10010004

✓ Data ✓ Text

```
Address     Value (+0)  Value (+4)  Value (+8)  Value (+c)  Value (+10)  Value (+14)  Value (+18)  Value (+1c)
0x10010000   268697600  1819043144  1867980911   174353522            0            0            0            0
0x10010020           0           0           0           0            0            0            0            0
0x10010040           0           0           0           0            0            0            0            0
0x10010060           0           0           0           0            0            0            0            0
0x10010080           0           0           0           0            0            0            0            0
0x100100a0           0           0           0           0            0            0            0            0
0x100100c0           0           0           0           0            0            0            0            0
```

0x10010000 (.data) ✓ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

CSUN CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP122

MIPS Assembly

MIPS Lab 1

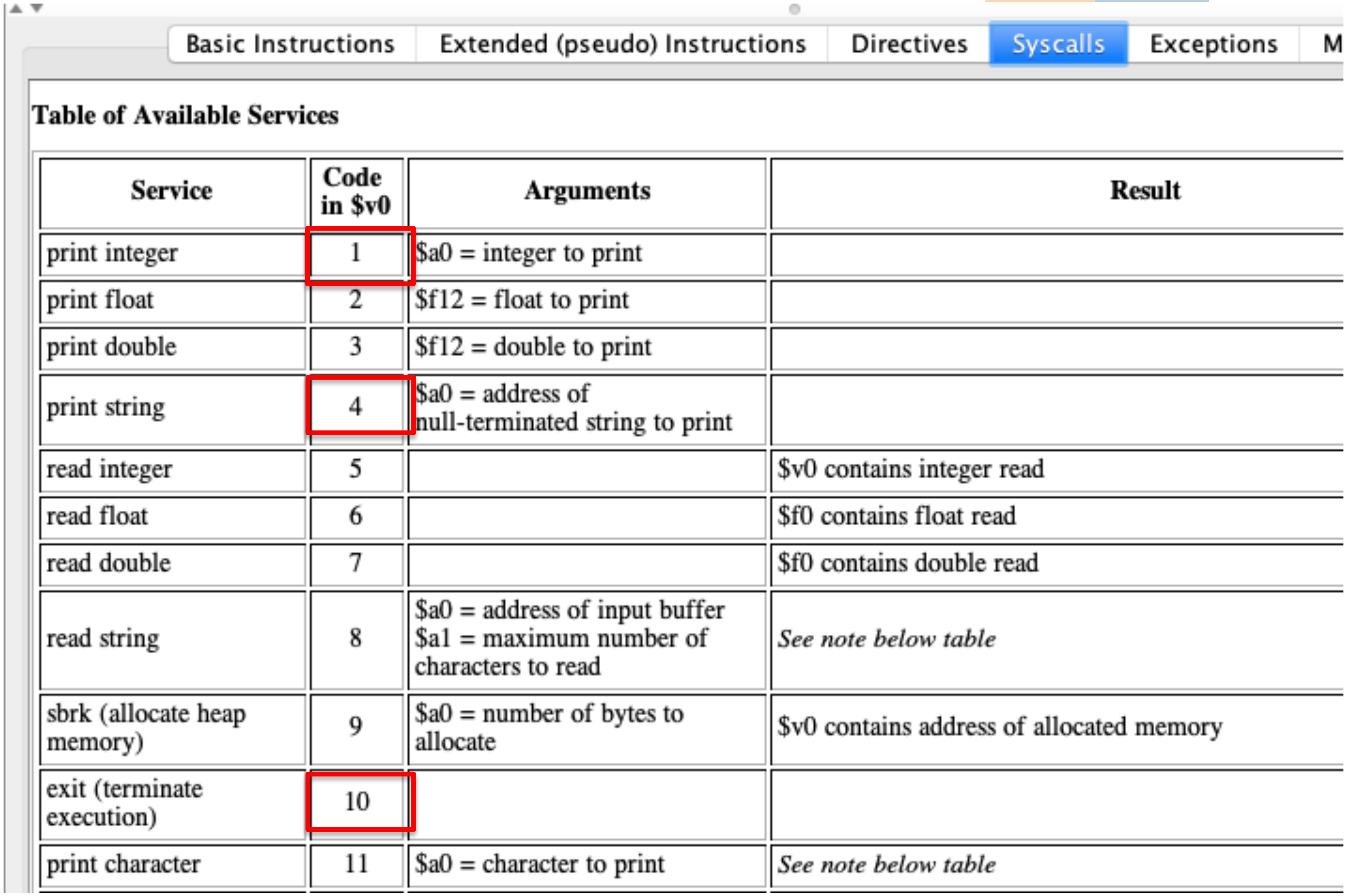


Table of Available Services

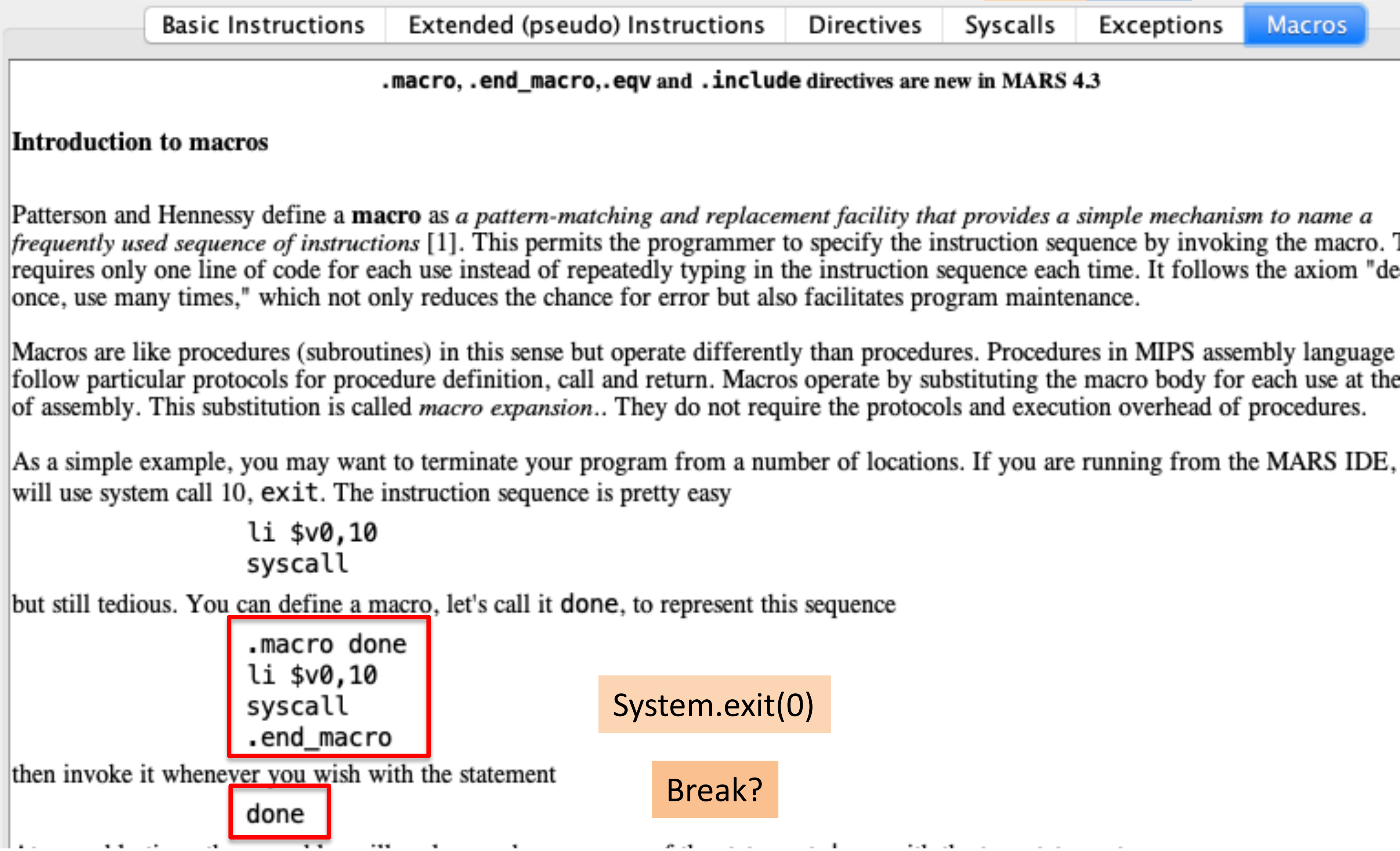
Service	Code in $v0	Arguments	Result
print integer	1	$a0 = integer to print	
print float	2	$f12 = float to print	
print double	3	$f12 = double to print	
print string	4	$a0 = address of null-terminated string to print	
read integer	5		$v0 contains integer read
read float	6		$f0 contains float read
read double	7		$f0 contains double read
read string	8	$a0 = address of input buffer $a1 = maximum number of characters to read	*See note below table*
sbrk (allocate heap memory)	9	$a0 = number of bytes to allocate	$v0 contains address of allocated memory
exit (terminate execution)	10		
print character	11	$a0 = character to print	*See note below table*

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

MIPS Assembly

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023

MIPS Lab 1



`.macro`, `.end_macro`,`.eqv` and `.include` directives are new in MARS 4.3

Introduction to macros

Patterson and Hennessy define a **macro** as *a pattern-matching and replacement facility that provides a simple mechanism to name a frequently used sequence of instructions* [1]. This permits the programmer to specify the instruction sequence by invoking the macro. T
requires only one line of code for each use instead of repeatedly typing in the instruction sequence each time. It follows the axiom "de
once, use many times," which not only reduces the chance for error but also facilitates program maintenance.

Macros are like procedures (subroutines) in this sense but operate differently than procedures. Procedures in MIPS assembly language
follow particular protocols for procedure definition, call and return. Macros operate by substituting the macro body for each use at the
of assembly. This substitution is called *macro expansion*.. They do not require the protocols and execution overhead of procedures.

As a simple example, you may want to terminate your program from a number of locations. If you are running from the MARS IDE,
will use system call 10, `exit`. The instruction sequence is pretty easy

```
li $v0,10
syscall
```

but still tedious. You can define a macro, let's call it `done`, to represent this sequence

```
.macro done
li $v0,10
syscall
.end_macro
```

then invoke it whenever you wish with the statement

```
done
```

System.exit(0)

Break?

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

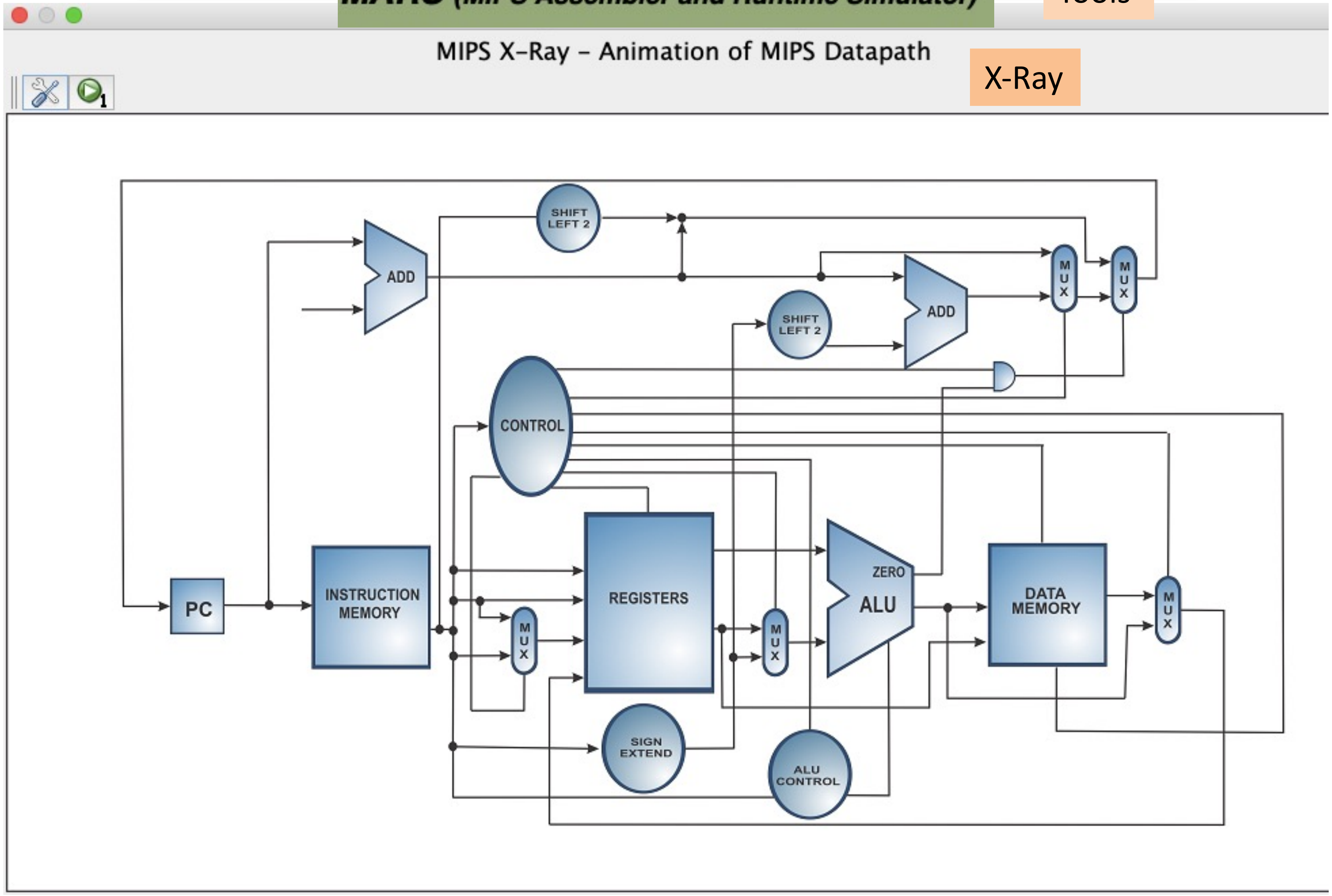
MARS

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman 2016-2023

MARS ***(MIPS Assembler and Runtime Simulator)***

Tools

X-Ray



CSUN CALIFORNIA STATE UNIVERSITY NORTHRIDGE

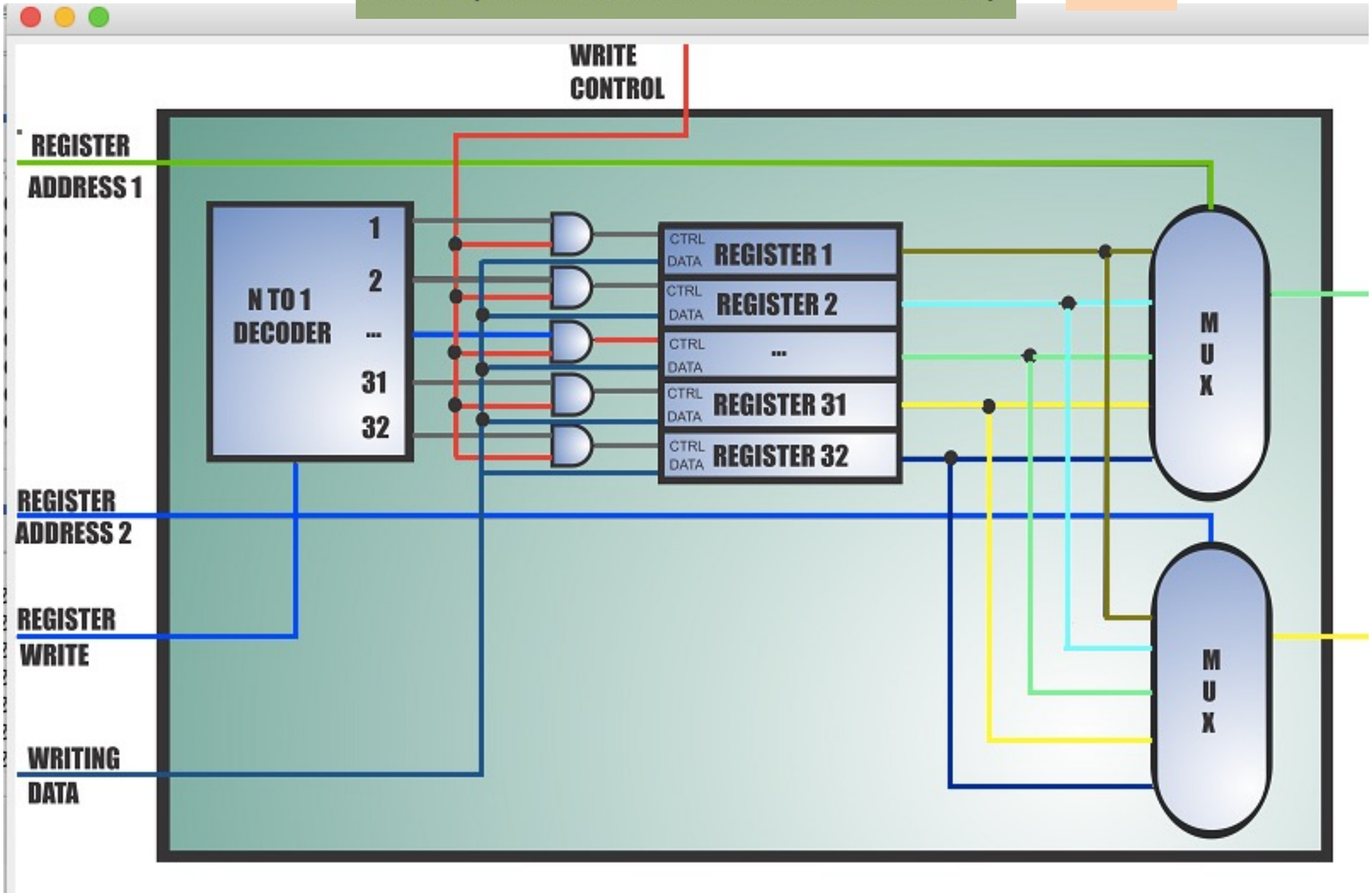
COMP122

MARS

DR JEFF SOFTWARE INDIE APP DEVELOPER

MARS ***(MIPS Assembler and Runtime Simulator)***

Tools



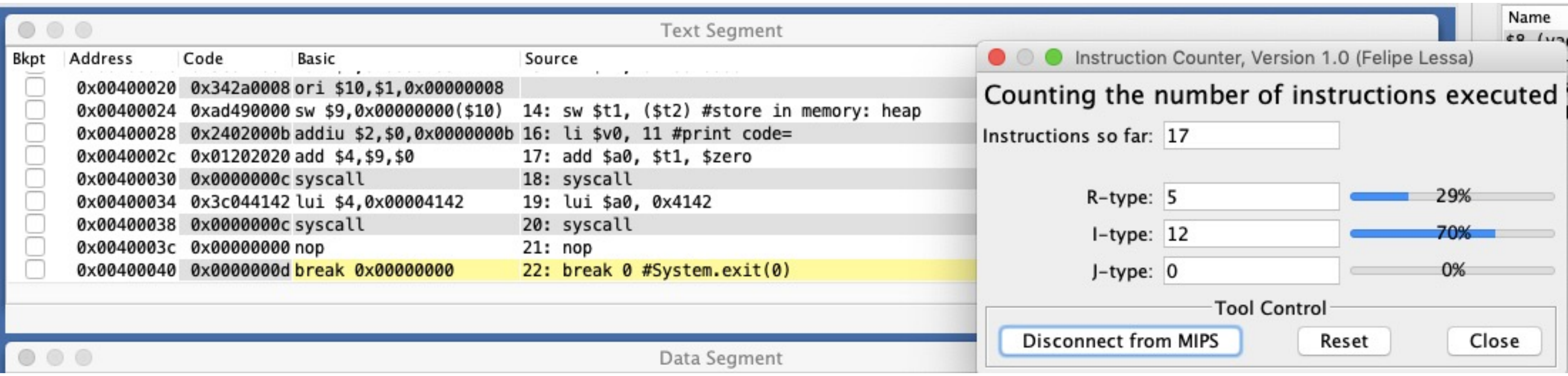
CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

MARS

MARS (MIPS Assembler and Runtime Simulator)

Tools

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman 2016-2023



```
Bkpt  Address     Code        Basic                         Source
      0x00400020  0x342a0008  ori $10,$1,0x00000008
      0x00400024  0xad490000  sw $9,0x00000000($10)   14:  sw $t1, ($t2) #store in memory: heap
      0x00400028  0x2402000b  addiu $2,$0,0x0000000b  16:  li $v0, 11 #print code=
      0x0040002c  0x01202020  add $4,$9,$0            17:  add $a0, $t1, $zero
      0x00400030  0x0000000c  syscall                 18:  syscall
      0x00400034  0x3c044142  lui $4,0x00004142       19:  lui $a0, 0x4142
      0x00400038  0x0000000c  syscall                 20:  syscall
      0x0040003c  0x00000000  nop                     21:  nop
      0x00400040  0x0000000d  break 0x00000000        22:  break 0 #System.exit(0)
```

Counting the number of instructions executed

Instructions so far: 17

R-type: 5 29%
I-type: 12 70%
J-type: 0 0%

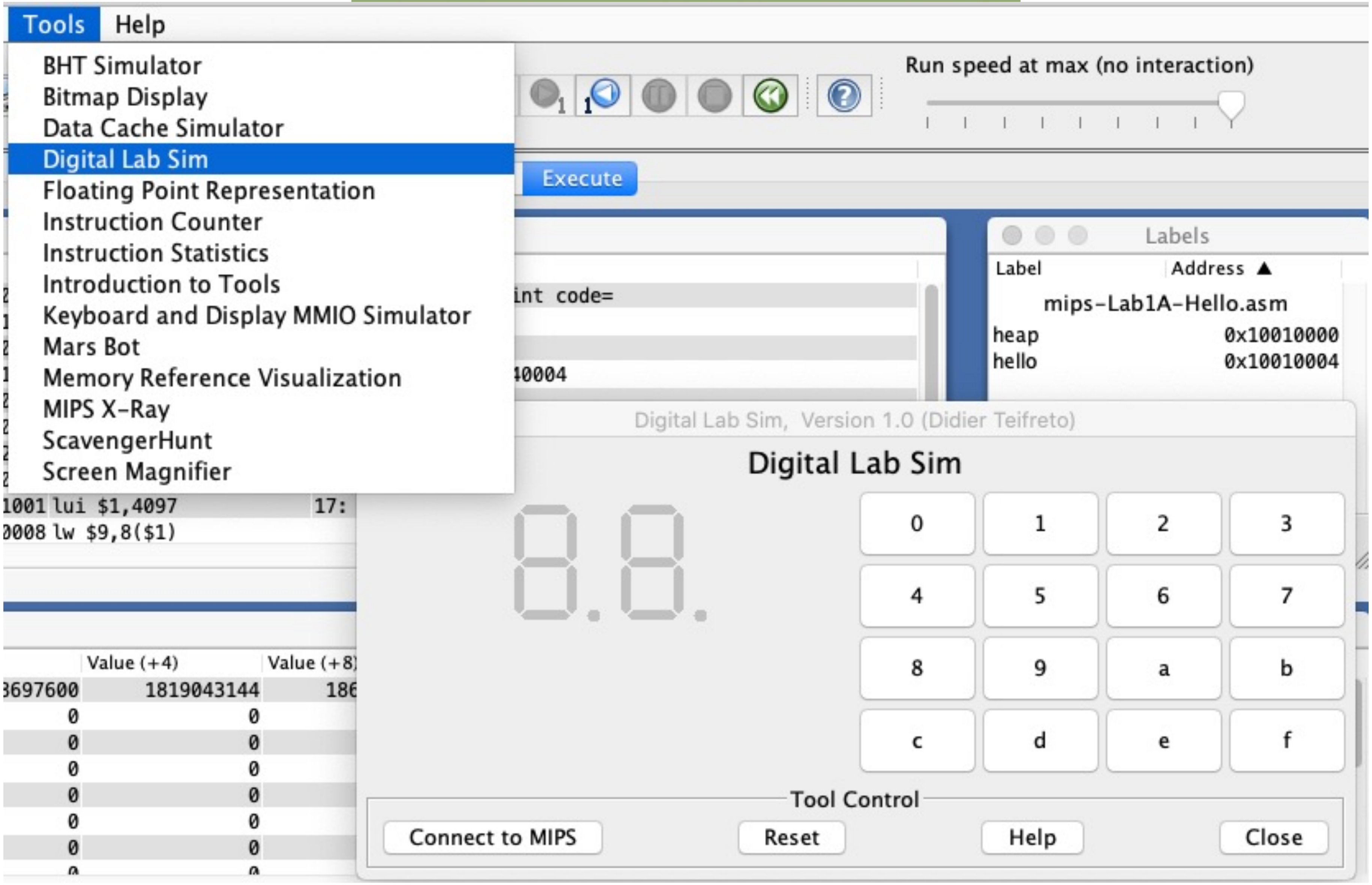
CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

MARS

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023

MARS ***(MIPS Assembler and Runtime Simulator)***

Tools



```
int code=
40004
1001 lui $1,4097        17:
0008 lw  $9,8($1)
```

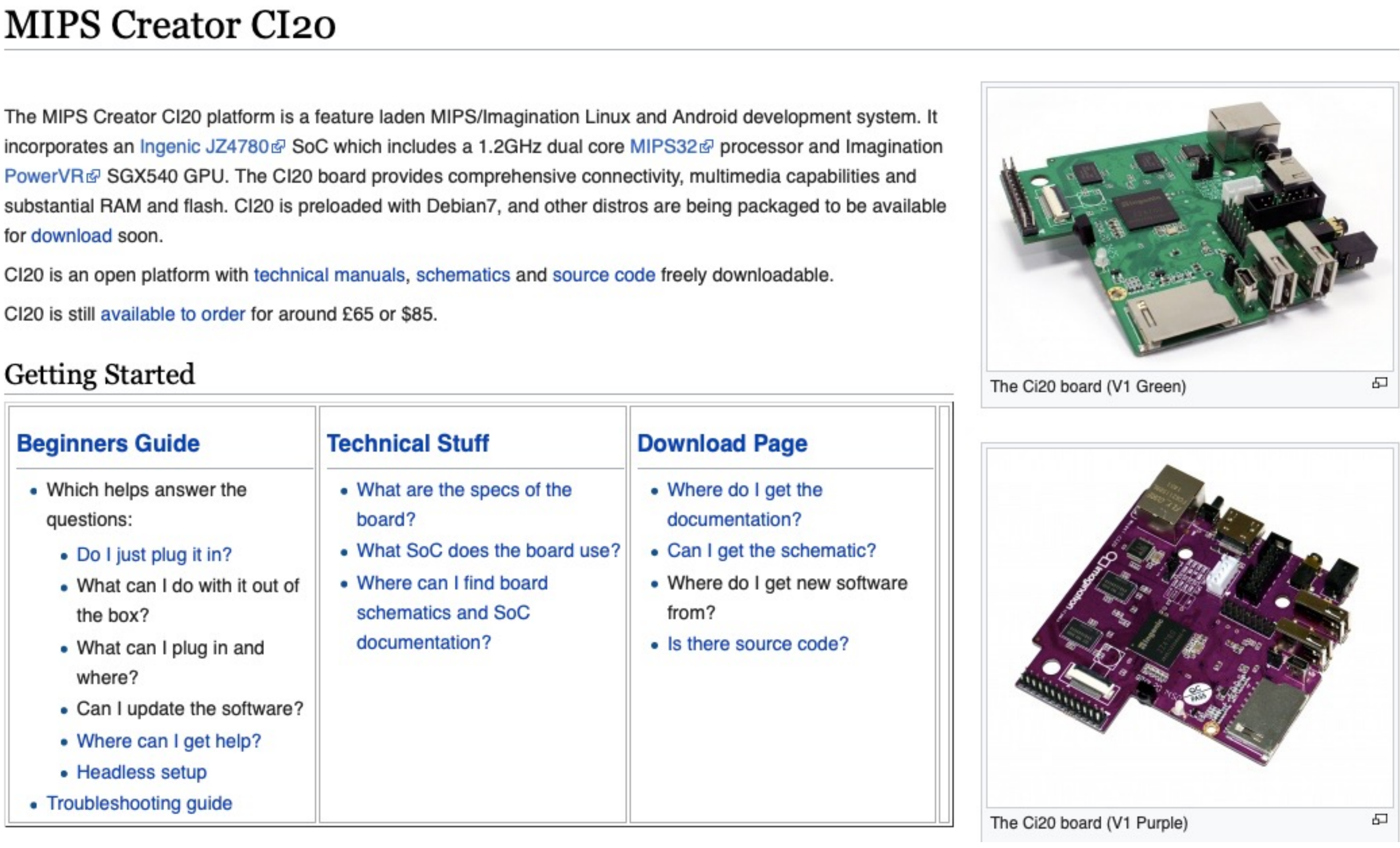
Label	Address ▲
mips-Lab1A-Hello.asm	
heap	0x10010000
hello	0x10010004

```
         Value (+4)   Value (+8)
3697600  1819043144   186
      0           0
      0           0
      0           0
      0           0
      0           0
      0           0
```

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

MIPS Dev Board

DR JEFF SOFTWARE
INDIE APP DEVELOPER



MIPS Creator CI20

The MIPS Creator CI20 platform is a feature laden MIPS/Imagination Linux and Android development system. It incorporates an Ingenic JZ4780 SoC which includes a 1.2GHz dual core MIPS32 processor and Imagination PowerVR SGX540 GPU. The CI20 board provides comprehensive connectivity, multimedia capabilities and substantial RAM and flash. CI20 is preloaded with Debian7, and other distros are being packaged to be available for download soon.

CI20 is an open platform with technical manuals, schematics and source code freely downloadable.

CI20 is still available to order for around £65 or $85.

The Ci20 board (V1 Green)

Getting Started

Beginners Guide

- Which helps answer the questions:
  - Do I just plug it in?
  - What can I do with it out of the box?
  - What can I plug in and where?
  - Can I update the software?
  - Where can I get help?
  - Headless setup
- Troubleshooting guide

Technical Stuff

- What are the specs of the board?
- What SoC does the board use?
- Where can I find board schematics and SoC documentation?

Download Page

- Where do I get the documentation?
- Can I get the schematic?
- Where do I get new software from?
- Is there source code?

The Ci20 board (V1 Purple)

COMP122

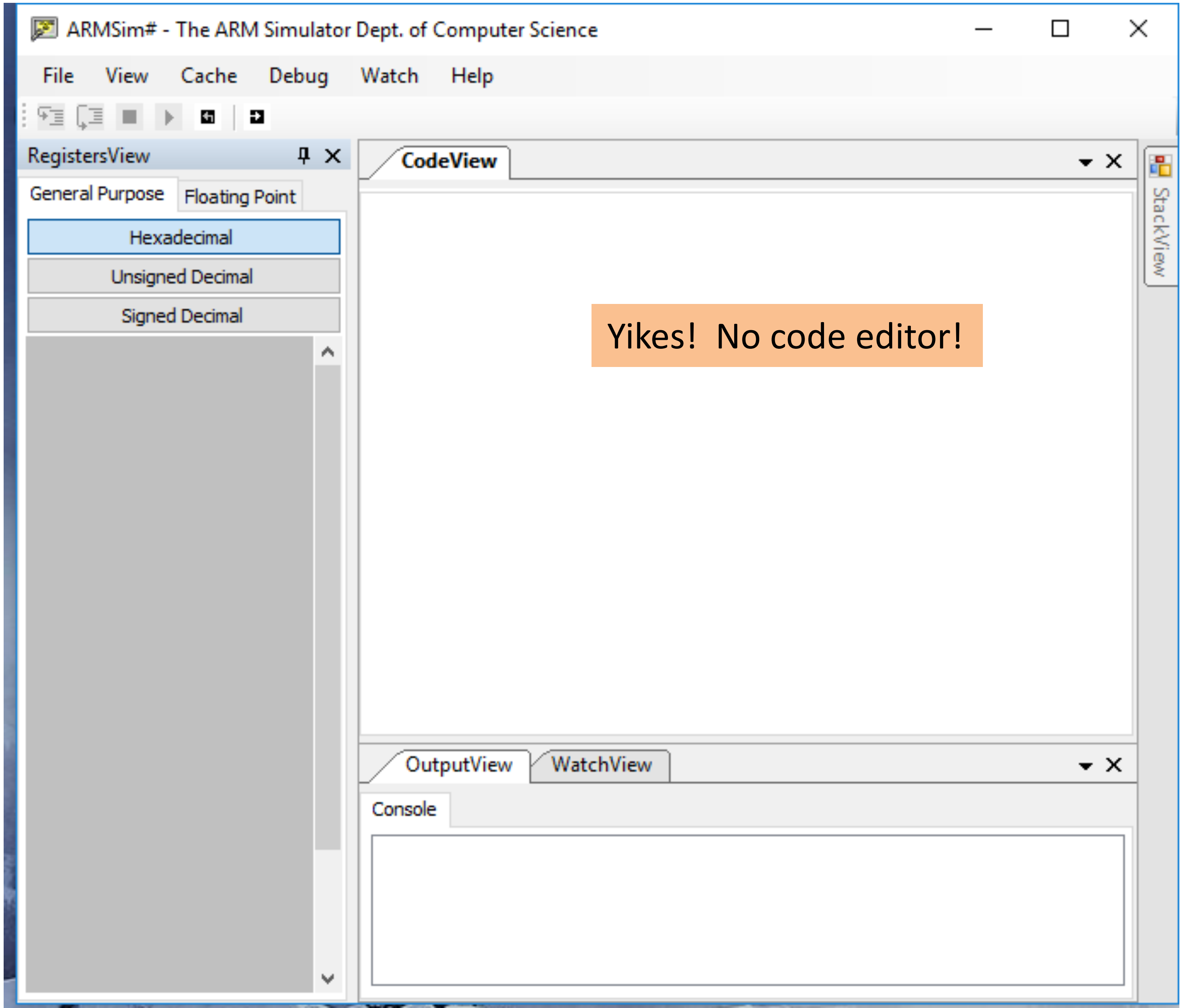
Lab

ARM Sim

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

ARM Sim

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023



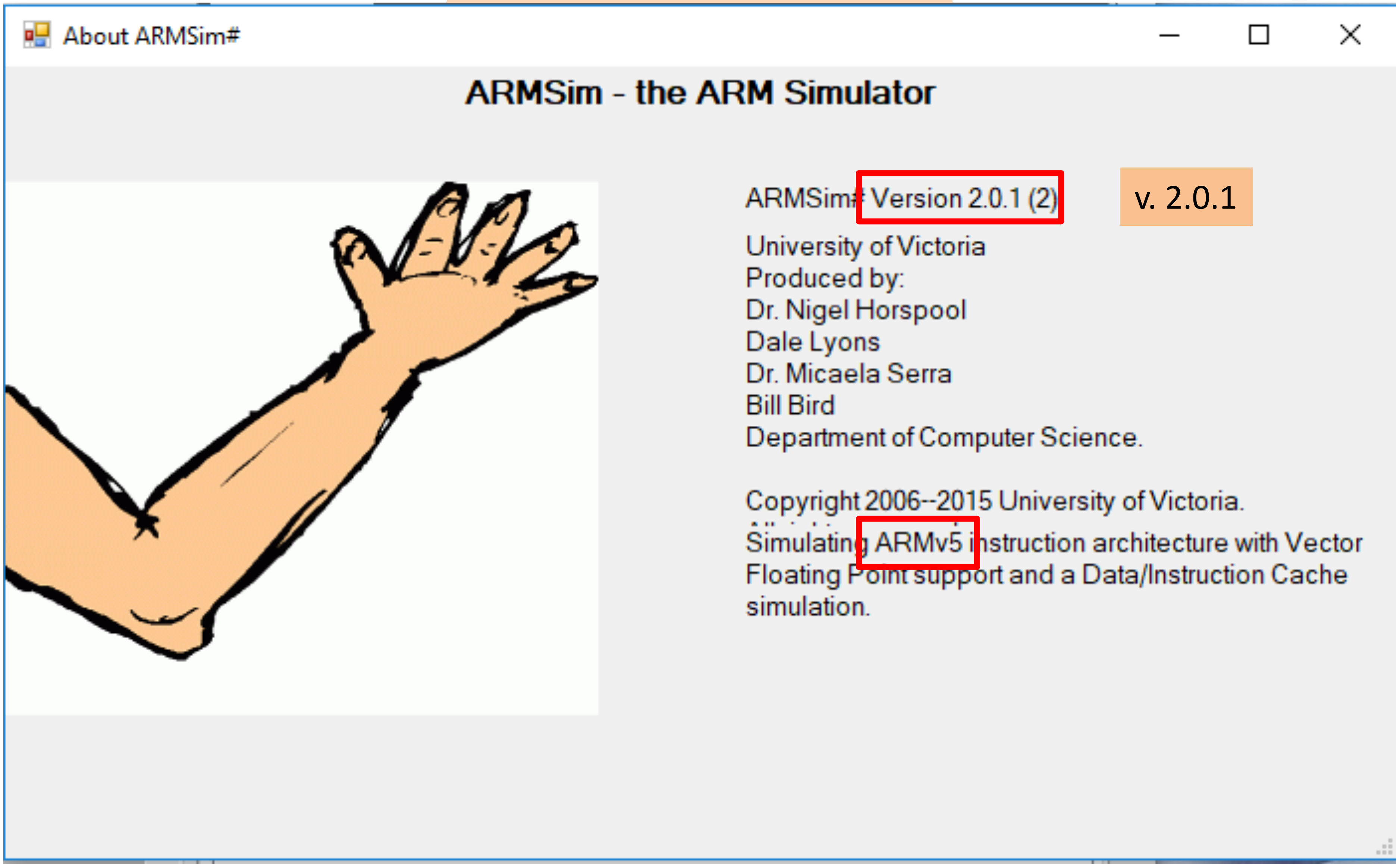
Yikes! No code editor!

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

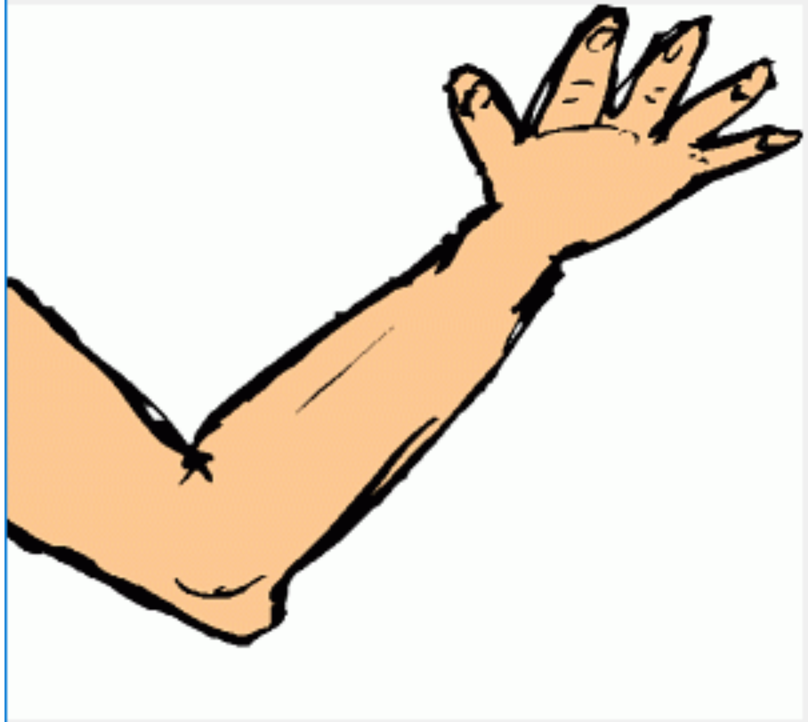
ARM Sim

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023

tinyurl.com/armsimcsun


About ARMSim#

ARMSim - the ARM Simulator



ARMSim# Version 2.0.1 (2)

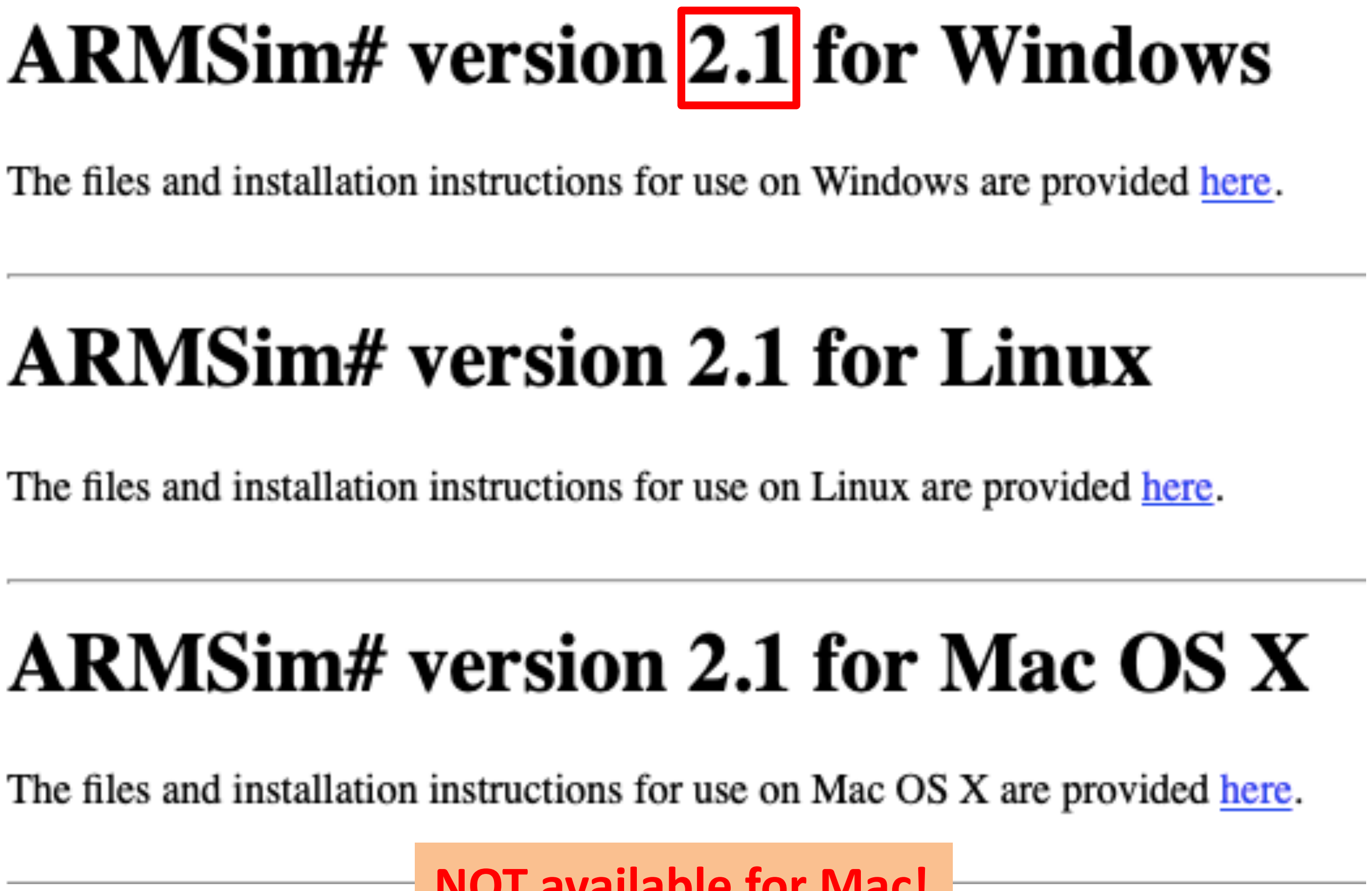
v. 2.0.1

University of Victoria
Produced by:
Dr. Nigel Horspool
Dale Lyons
Dr. Micaela Serra
Bill Bird
Department of Computer Science.

Copyright 2006--2015 University of Victoria.
Simulating ARMv5 instruction architecture with Vector Floating Point support and a Data/Instruction Cache simulation.

ARM Sim

tinyurl.com/armsimcsun



ARMSim# version 2.1 for Windows

The files and installation instructions for use on Windows are provided here.

ARMSim# version 2.1 for Linux

The files and installation instructions for use on Linux are provided here.

ARMSim# version 2.1 for Mac OS X

The files and installation instructions for use on Mac OS X are provided here.

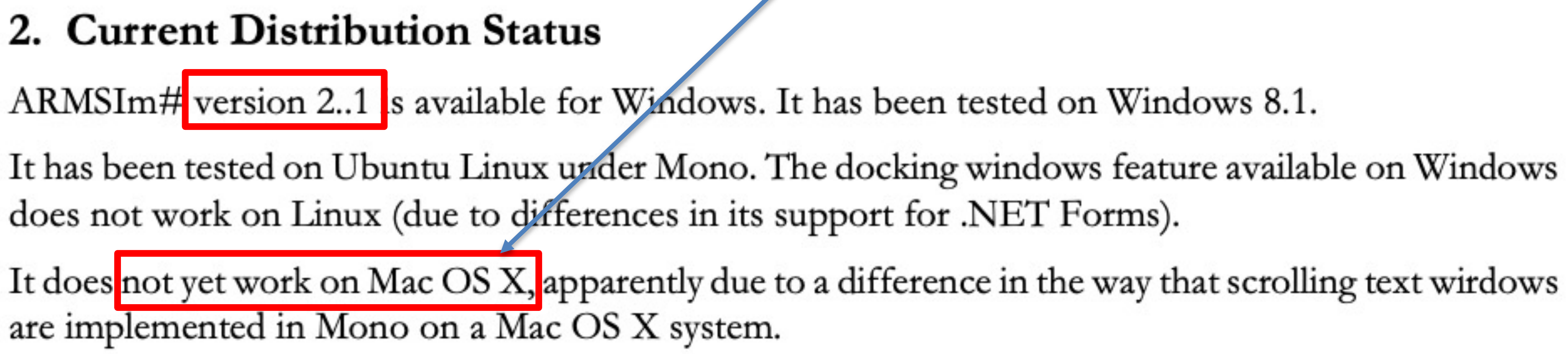
NOT available for Mac!

CSUN
CALIFORNIA
STATE UNIVERSITY
NORTHRIDGE
COMP122

Mac OS X

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023

NOT available for Mac!



2. Current Distribution Status

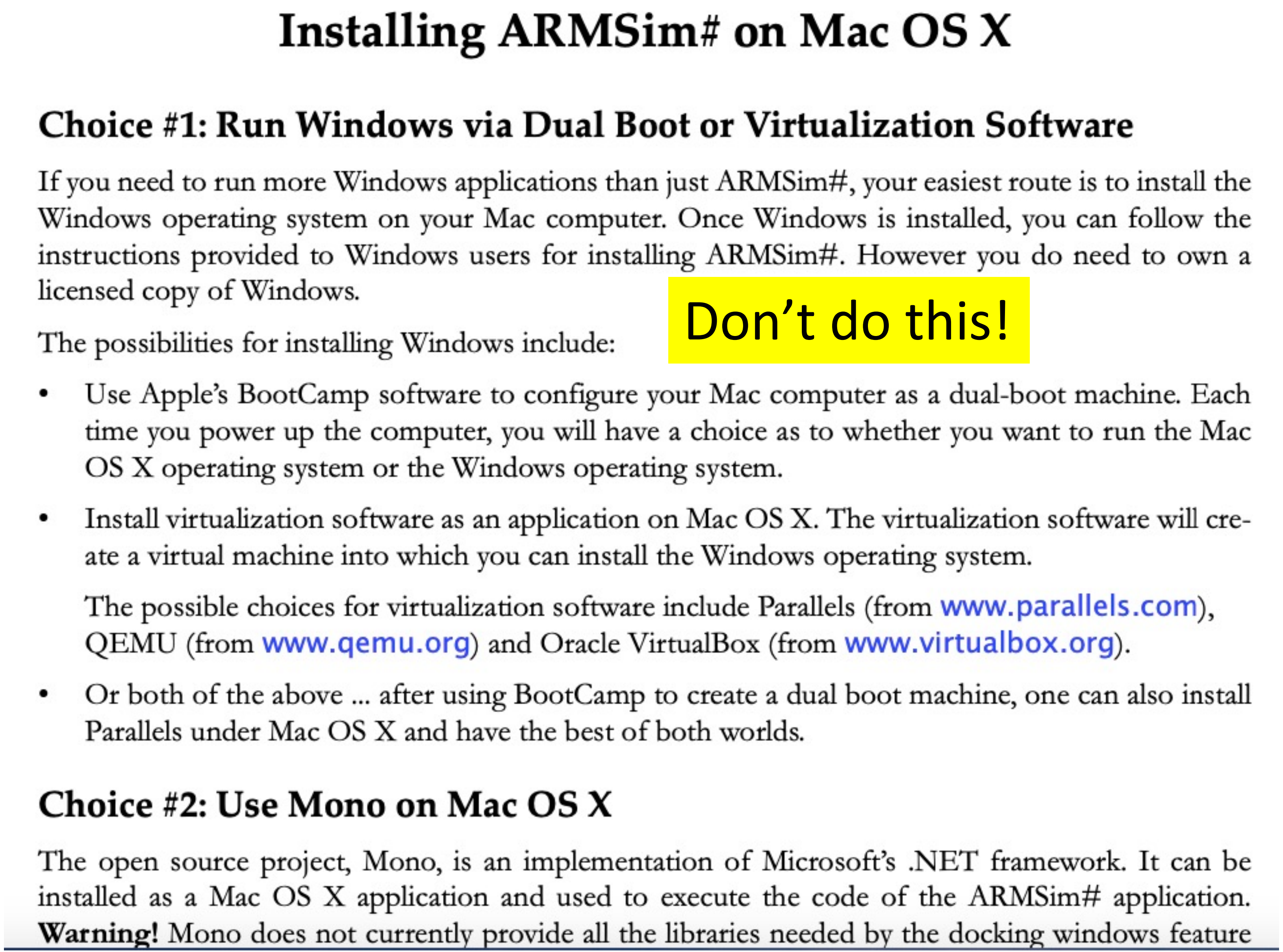
ARMSIm# version 2..1 is available for Windows. It has been tested on Windows 8.1.

It has been tested on Ubuntu Linux under Mono. The docking windows feature available on Windows does not work on Linux (due to differences in its support for .NET Forms).

It does not yet work on Mac OS X, apparently due to a difference in the way that scrolling text windows are implemented in Mono on a Mac OS X system.

COMP122

Mac OS X



Installing ARMSim# on Mac OS X

Choice #1: Run Windows via Dual Boot or Virtualization Software

If you need to run more Windows applications than just ARMSim#, your easiest route is to install the Windows operating system on your Mac computer. Once Windows is installed, you can follow the instructions provided to Windows users for installing ARMSim#. However you do need to own a licensed copy of Windows.

Don't do this!

The possibilities for installing Windows include:

- Use Apple's BootCamp software to configure your Mac computer as a dual-boot machine. Each time you power up the computer, you will have a choice as to whether you want to run the Mac OS X operating system or the Windows operating system.
- Install virtualization software as an application on Mac OS X. The virtualization software will create a virtual machine into which you can install the Windows operating system.

  The possible choices for virtualization software include Parallels (from www.parallels.com), QEMU (from www.qemu.org) and Oracle VirtualBox (from www.virtualbox.org).
- Or both of the above ... after using BootCamp to create a dual boot machine, one can also install Parallels under Mac OS X and have the best of both worlds.

Choice #2: Use Mono on Mac OS X

The open source project, Mono, is an implementation of Microsoft's .NET framework. It can be installed as a Mac OS X application and used to execute the code of the ARMSim# application. **Warning!** Mono does not currently provide all the libraries needed by the docking windows feature

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

Virtual Windows 11 on Mac

DR JEFF SOFTWARE
INDIE APP DEVELOPER

I ran Windows 11 on an M1 Mac: Here's my experience

By Shubham Agarwal published 27 days ago

We put Parallels' Windows 11 virtual machine to the test on an M1 Mac Mini to see if it's worth your money.

Windows 11 on a Mac (Image credit: Parallels)

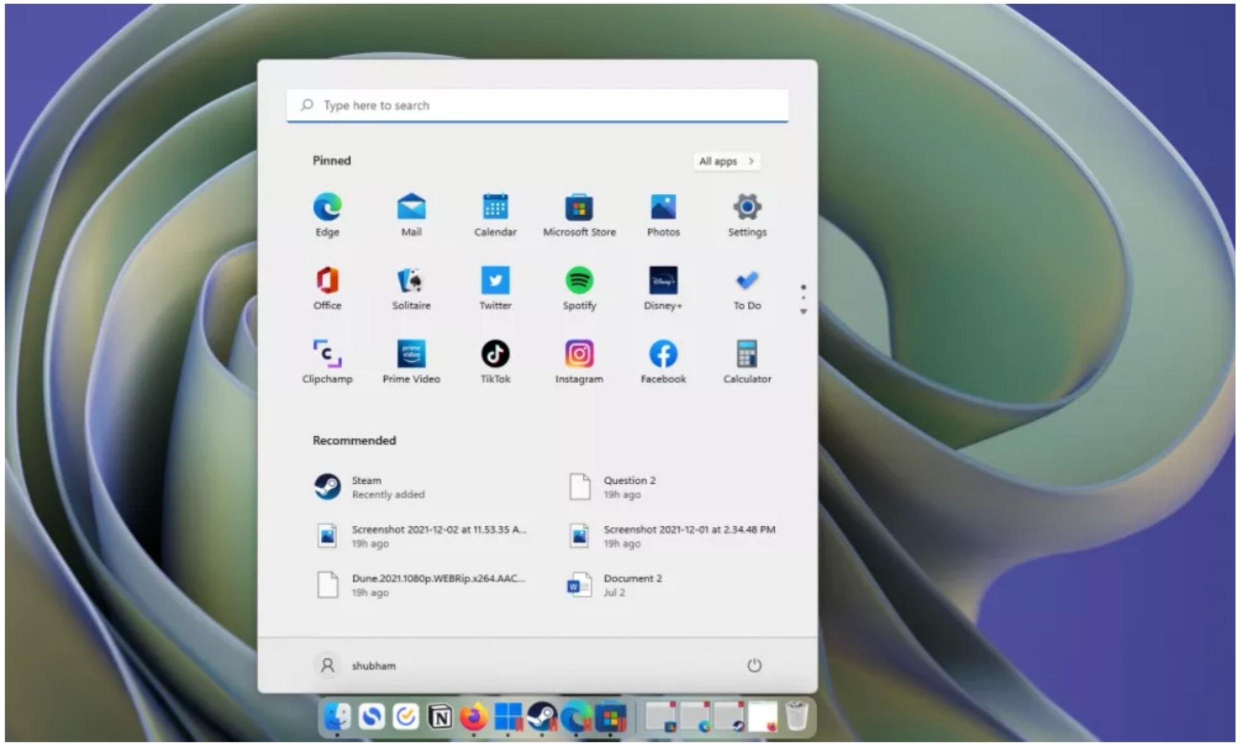
CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

Virtual Windows 11 on Mac

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman 2016-2023

How stable is a Windows 11 virtual machine on an M1 Mac?

Windows 11 on Parallels was remarkably stable for a virtual machine, and that allowed me to use it as my primary workspace.



Windows 11 on a Mac (Image credit: Parallels)

Everything from Windows 11's refreshed animations to resource-intensive multitasking worked as Microsoft intended it to. Plus, it can wake up from sleep

CSUN CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP122

ARMsim 2.1

DR JEFF SOFTWARE INDIE APP DEVELOPER

Introducing ARMSim# Version 2.1

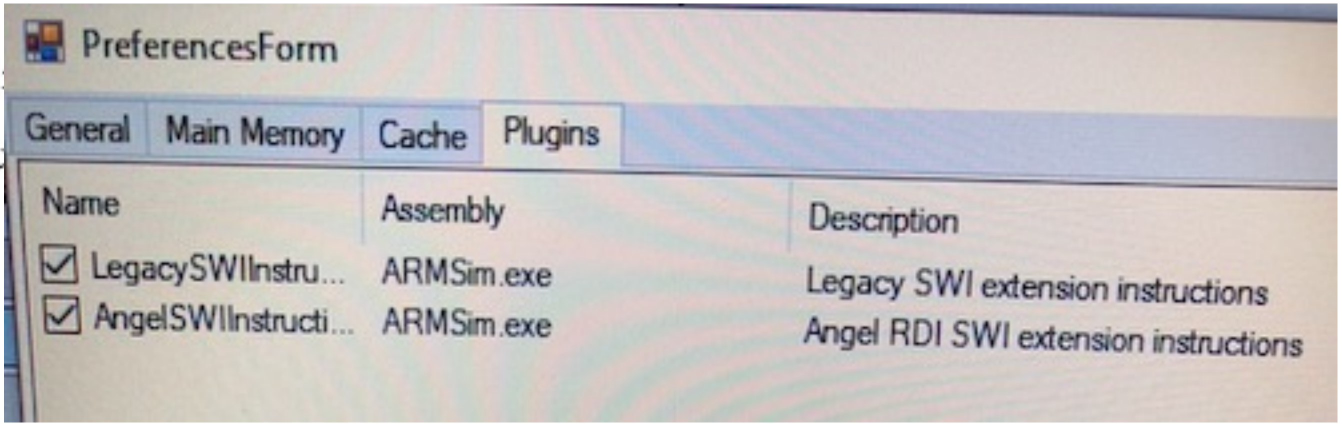
1. What is Different?

Version 2.1 is a major re-design of ARMSim# in three main respects:

1. Instead of parsing and assembling ARM source code itself, ARMSim# now invokes the Gnu Assembler program `as` to perform the task.
2. Instead of using a set of extended SWI instructions based on the ARM RDI family to perform I/O and other system tasks, a new set known as the Angel SWI instructions has been adopted as the default set.
3. The undocumented support for scripting has been replaced by an extended set of command-line options.

Each of these

Some tidying u


PreferencesForm

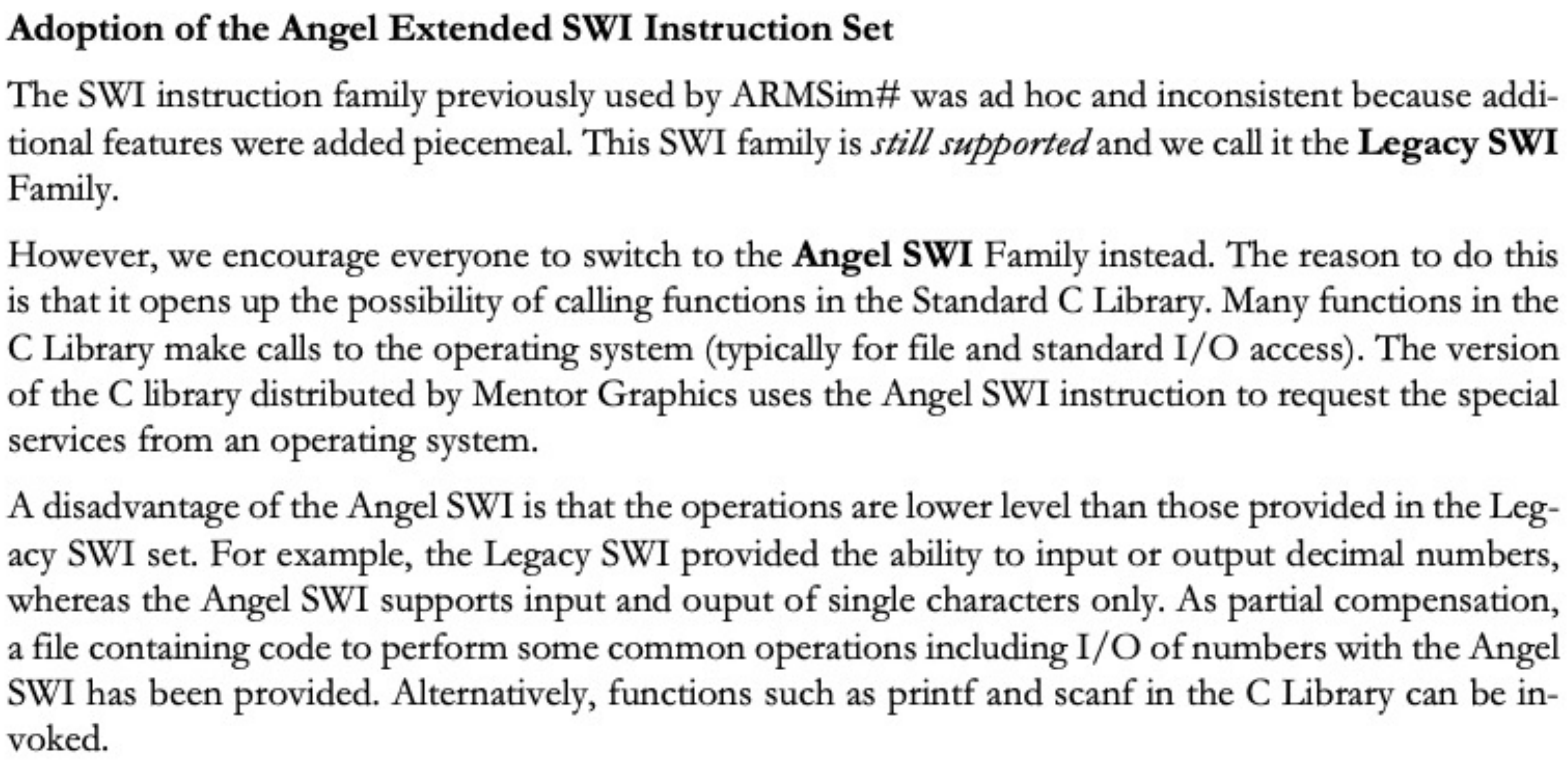
Name	Assembly	Description
☑ LegacySWIInstru...	ARMSim.exe	Legacy SWI extension instructions
☑ AngelSWIInstructi...	ARMSim.exe	Angel RDI SWI extension instructions

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

ARMsim 2.1

Angel SWI

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman 2016-2023



Adoption of the Angel Extended SWI Instruction Set

The SWI instruction family previously used by ARMSim# was ad hoc and inconsistent because additional features were added piecemeal. This SWI family is *still supported* and we call it the **Legacy SWI** Family.

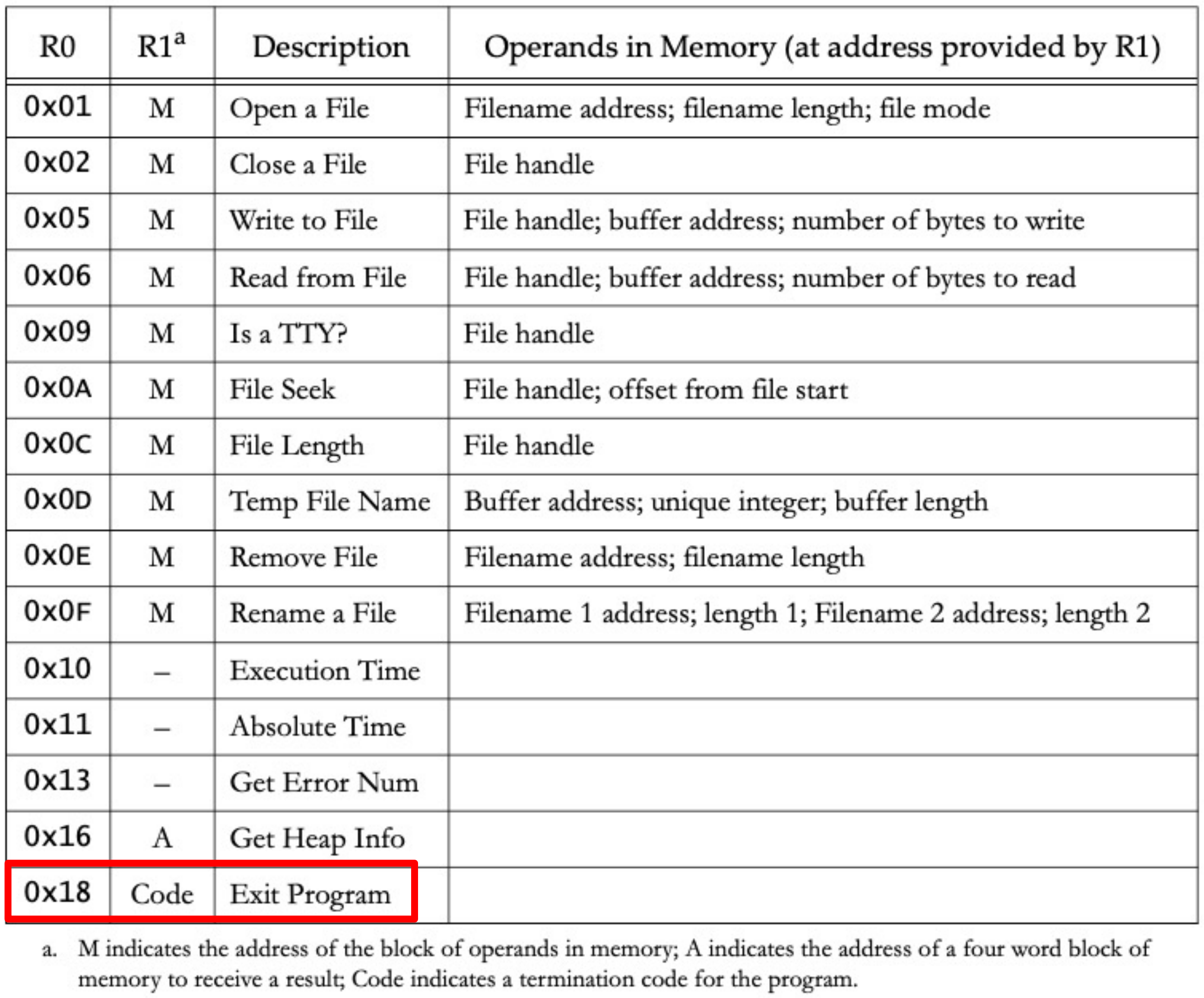
However, we encourage everyone to switch to the **Angel SWI** Family instead. The reason to do this is that it opens up the possibility of calling functions in the Standard C Library. Many functions in the C Library make calls to the operating system (typically for file and standard I/O access). The version of the C library distributed by Mentor Graphics uses the Angel SWI instruction to request the special services from an operating system.

A disadvantage of the Angel SWI is that the operations are lower level than those provided in the Legacy SWI set. For example, the Legacy SWI provided the ability to input or output decimal numbers, whereas the Angel SWI supports input and ouput of single characters only. As partial compensation, a file containing code to perform some common operations including I/O of numbers with the Angel SWI has been provided. Alternatively, functions such as printf and scanf in the C Library can be invoked.

ARMsim 2.1

Angel SWI

Table 1: Summary of Angel SWI Operations



R0	R1[a]	Description	Operands in Memory (at address provided by R1)
0x01	M	Open a File	Filename address; filename length; file mode
0x02	M	Close a File	File handle
0x05	M	Write to File	File handle; buffer address; number of bytes to write
0x06	M	Read from File	File handle; buffer address; number of bytes to read
0x09	M	Is a TTY?	File handle
0x0A	M	File Seek	File handle; offset from file start
0x0C	M	File Length	File handle
0x0D	M	Temp File Name	Buffer address; unique integer; buffer length
0x0E	M	Remove File	Filename address; filename length
0x0F	M	Rename a File	Filename 1 address; length 1; Filename 2 address; length 2
0x10	–	Execution Time	
0x11	–	Absolute Time	
0x13	–	Get Error Num	
0x16	A	Get Heap Info	
0x18	Code	Exit Program	

a. M indicates the address of the block of operands in memory; A indicates the address of a four word block of memory to receive a result; Code indicates a termination code for the program.

ARMsim 2.1

Angel SWI

```
@    Example of using the Angel SWI operations

         ...                            @ omitted code

         ldr       R1, =OpenParams      @ parameters block for OPEN
         mov       R0, #0x01            @ code number for Open File
         swi       0x123456             @ open a text file for input
         cmp       R0, #0
         blt       OpenError            @ branch if there was an error
         ldr       R1, =ReadParams
         str       R0,[R1]              @ save the file handle into
                                        @   parameters block for READ
         mov       R0, #0x06            @ code number for Read File
         swi       0x123456             @ read from the text file
         cmp       R0, #0
         bne       ReadError            @ branch if there was an error
```

ARM Sim

Assembly Manual

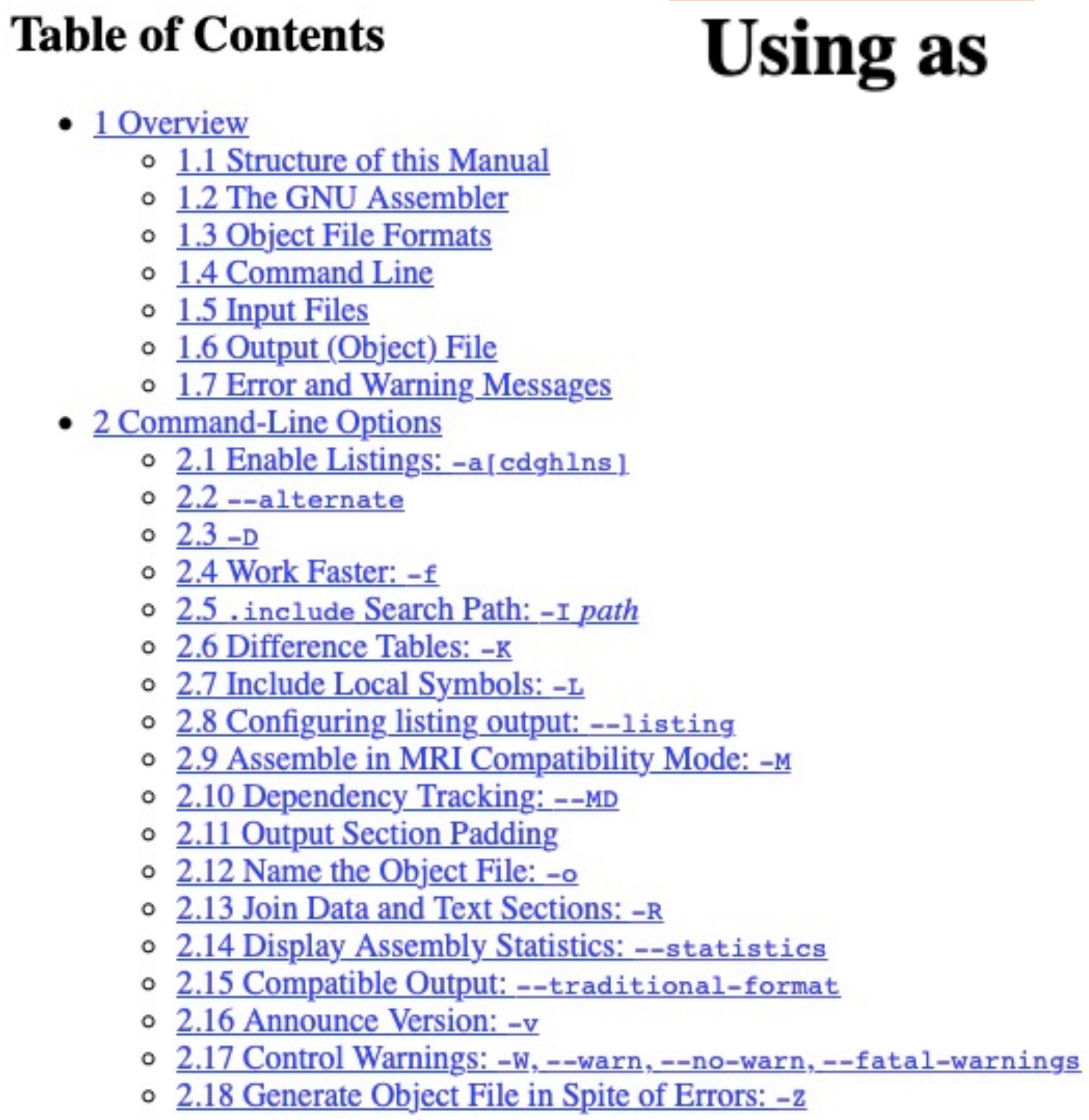
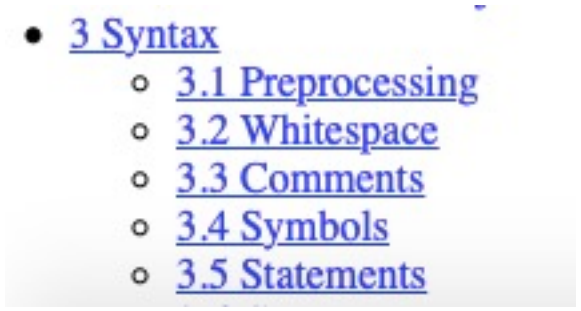


Table of Contents

Using as



CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

Lab

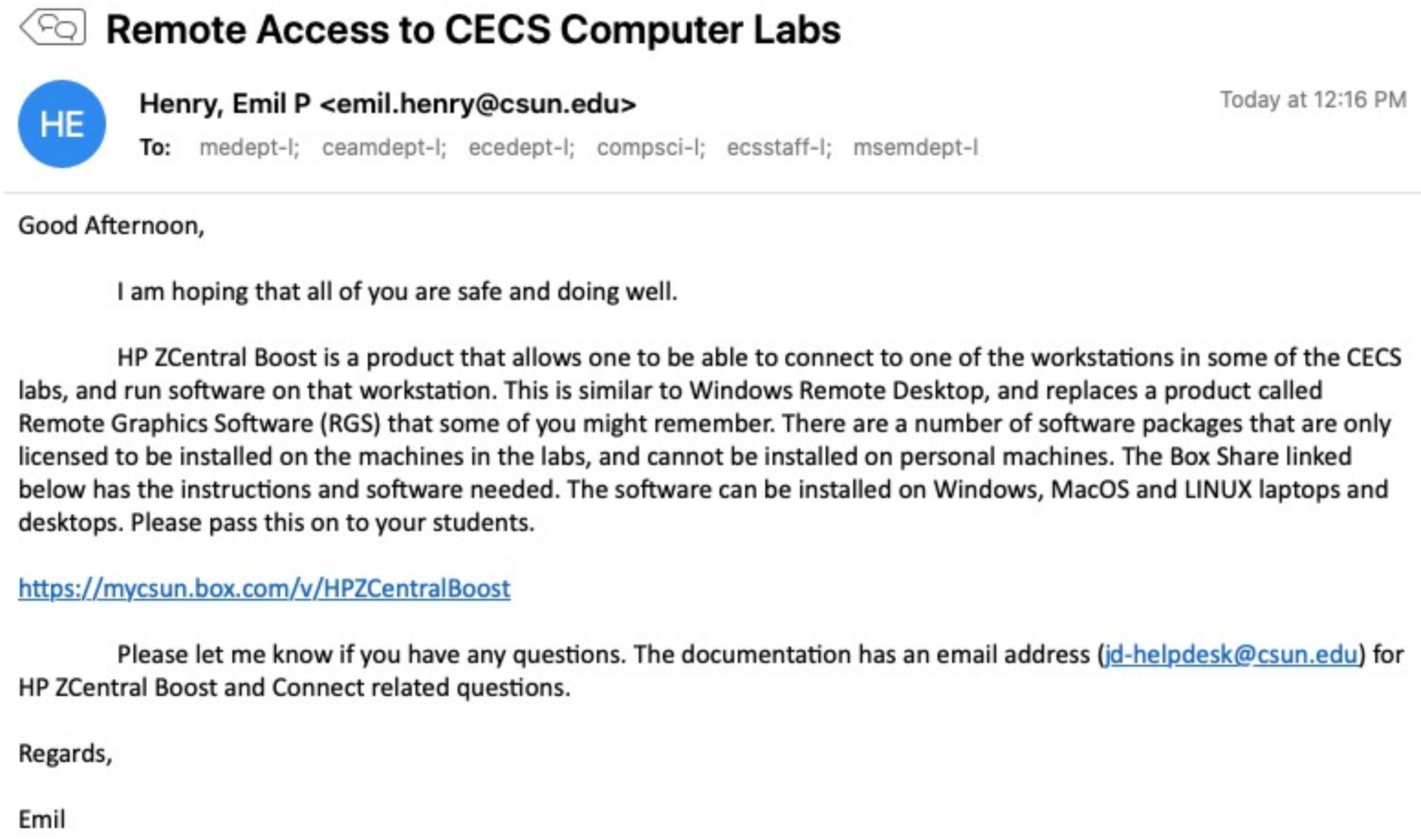
DR JEFF SOFTWARE
INDIE APP DEVELOPER

CECS IT Labs

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

CECS Labs

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023



Remote Access to CECS Computer Labs

HE **Henry, Emil P <emil.henry@csun.edu>** Today at 12:16 PM

To: medept-l; ceamdept-l; ecedept-l; compsci-l; ecsstaff-l; msemdept-l

Good Afternoon,

I am hoping that all of you are safe and doing well.

HP ZCentral Boost is a product that allows one to be able to connect to one of the workstations in some of the CECS labs, and run software on that workstation. This is similar to Windows Remote Desktop, and replaces a product called Remote Graphics Software (RGS) that some of you might remember. There are a number of software packages that are only licensed to be installed on the machines in the labs, and cannot be installed on personal machines. The Box Share linked below has the instructions and software needed. The software can be installed on Windows, MacOS and LINUX laptops and desktops. Please pass this on to your students.

https://mycsun.box.com/v/HPZCentralBoost

Please let me know if you have any questions. The documentation has an email address (jd-helpdesk@csun.edu) for HP ZCentral Boost and Connect related questions.

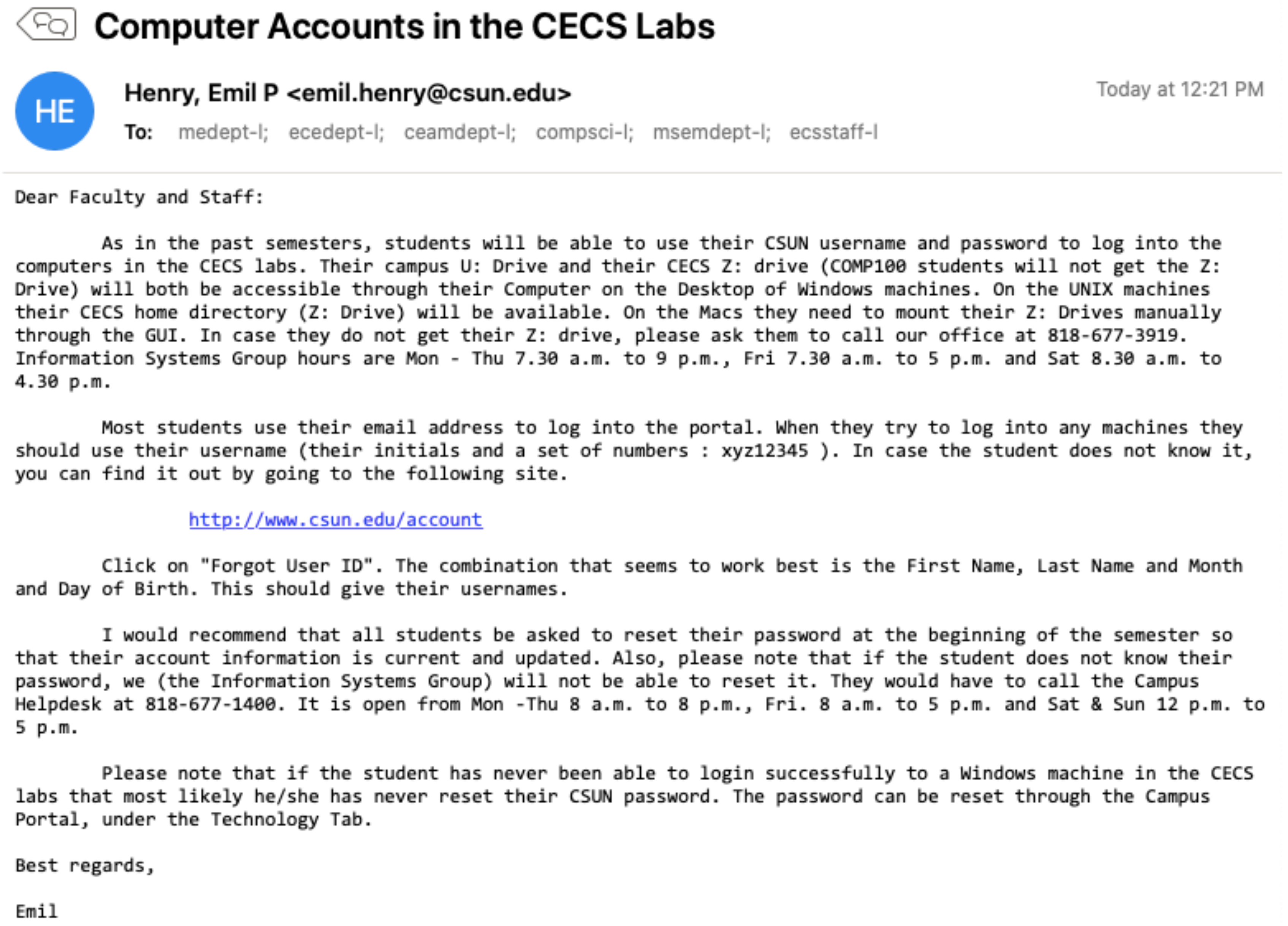
Regards,

Emil

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

CECS Labs

DR JEFF SOFTWARE
INDIE APP DEVELOPER
© Jeff Drobman
2016-2023



Computer Accounts in the CECS Labs

Henry, Emil P <emil.henry@csun.edu> Today at 12:21 PM

To: medept-l; ecedept-l; ceamdept-l; compsci-l; msemdept-l; ecsstaff-l

Dear Faculty and Staff:

As in the past semesters, students will be able to use their CSUN username and password to log into the computers in the CECS labs. Their campus U: Drive and their CECS Z: drive (COMP100 students will not get the Z: Drive) will both be accessible through their Computer on the Desktop of Windows machines. On the UNIX machines their CECS home directory (Z: Drive) will be available. On the Macs they need to mount their Z: Drives manually through the GUI. In case they do not get their Z: drive, please ask them to call our office at 818-677-3919. Information Systems Group hours are Mon - Thu 7.30 a.m. to 9 p.m., Fri 7.30 a.m. to 5 p.m. and Sat 8.30 a.m. to 4.30 p.m.

Most students use their email address to log into the portal. When they try to log into any machines they should use their username (their initials and a set of numbers : xyz12345). In case the student does not know it, you can find it out by going to the following site.

http://www.csun.edu/account

Click on "Forgot User ID". The combination that seems to work best is the First Name, Last Name and Month and Day of Birth. This should give their usernames.

I would recommend that all students be asked to reset their password at the beginning of the semester so that their account information is current and updated. Also, please note that if the student does not know their password, we (the Information Systems Group) will not be able to reset it. They would have to call the Campus Helpdesk at 818-677-1400. It is open from Mon -Thu 8 a.m. to 8 p.m., Fri. 8 a.m. to 5 p.m. and Sat & Sun 12 p.m. to 5 p.m.

Please note that if the student has never been able to login successfully to a Windows machine in the CECS labs that most likely he/she has never reset their CSUN password. The password can be reset through the Campus Portal, under the Technology Tab.

Best regards,

Emil

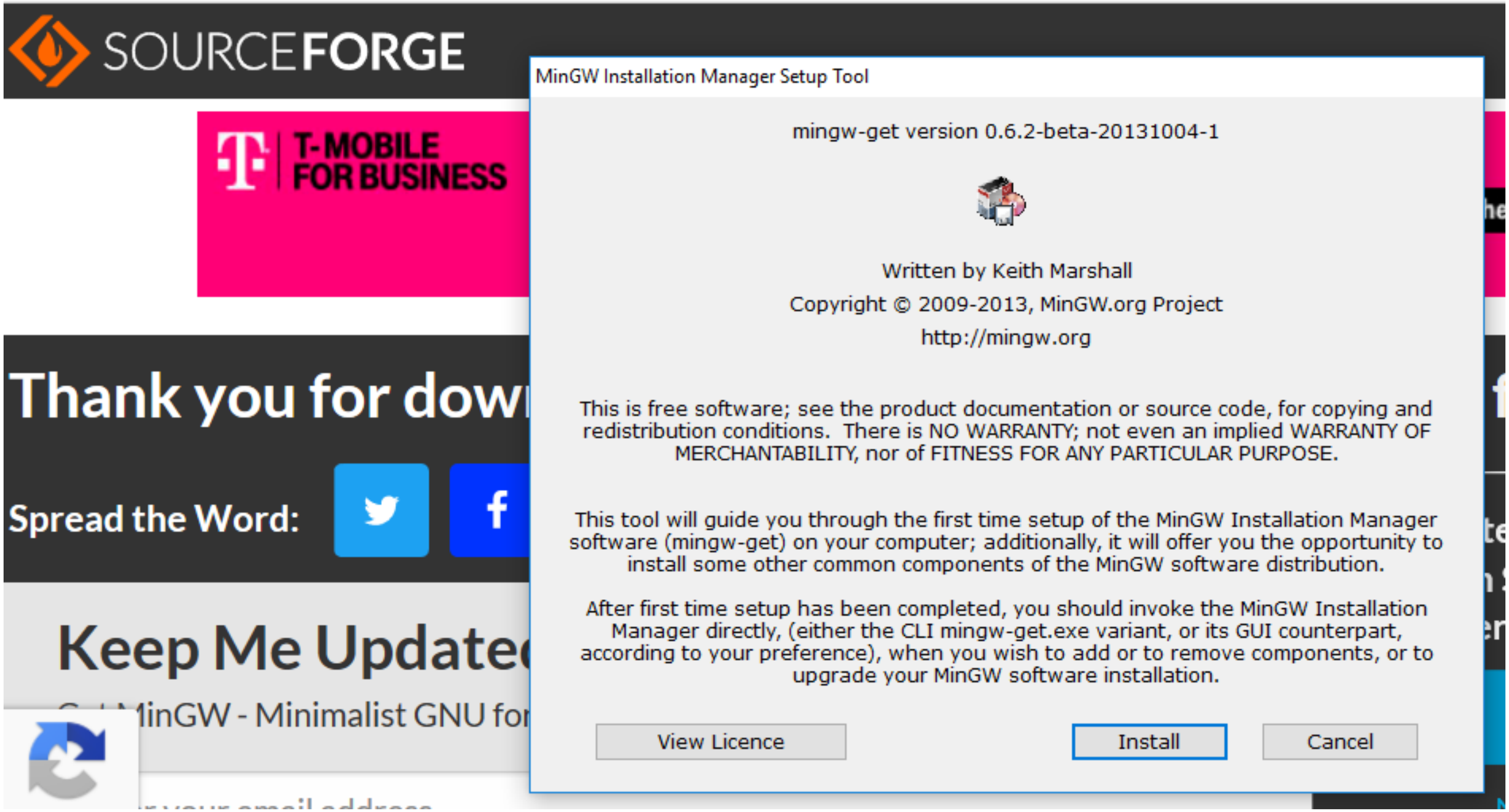
Lab

GNU

gcc

- C/C++ (.c/.cpp)
- Asm (.asm)

GNU - MinGW



MinGW Installation Manager Setup Tool

mingw-get version 0.6.2-beta-20131004-1

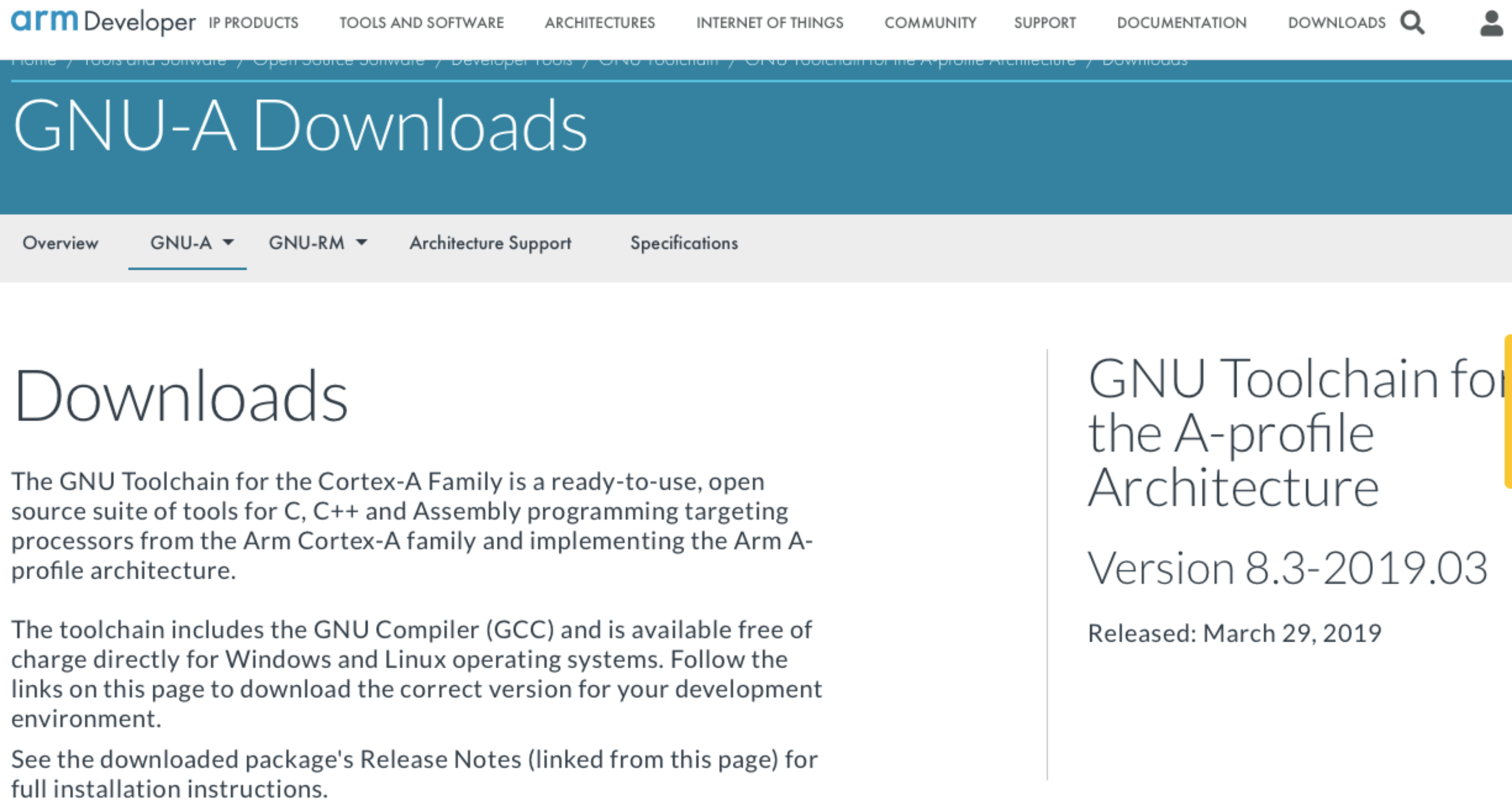
Written by Keith Marshall
Copyright © 2009-2013, MinGW.org Project
http://mingw.org

This is free software; see the product documentation or source code, for copying and redistribution conditions. There is NO WARRANTY; not even an implied WARRANTY OF MERCHANTABILITY, nor of FITNESS FOR ANY PARTICULAR PURPOSE.

This tool will guide you through the first time setup of the MinGW Installation Manager software (mingw-get) on your computer; additionally, it will offer you the opportunity to install some other common components of the MinGW software distribution.

After first time setup has been completed, you should invoke the MinGW Installation Manager directly, (either the CLI mingw-get.exe variant, or its GUI counterpart, according to your preference), when you wish to add or to remove components, or to upgrade your MinGW software installation.

ARM GNU-A



arm Developer IP PRODUCTS TOOLS AND SOFTWARE ARCHITECTURES INTERNET OF THINGS COMMUNITY SUPPORT DOCUMENTATION DOWNLOADS

Home / Tools and Software / Open Source Software / Developer Tools / GNU Toolchain / GNU Toolchain for the A-profile Architecture / Downloads

GNU-A Downloads

Overview GNU-A GNU-RM Architecture Support Specifications

Downloads

The GNU Toolchain for the Cortex-A Family is a ready-to-use, open source suite of tools for C, C++ and Assembly programming targeting processors from the Arm Cortex-A family and implementing the Arm A-profile architecture.

The toolchain includes the GNU Compiler (GCC) and is available free of charge directly for Windows and Linux operating systems. Follow the links on this page to download the correct version for your development environment.

See the downloaded package's Release Notes (linked from this page) for full installation instructions.

GNU Toolchain fo
the A-profile
Architecture

Version 8.3-2019.03

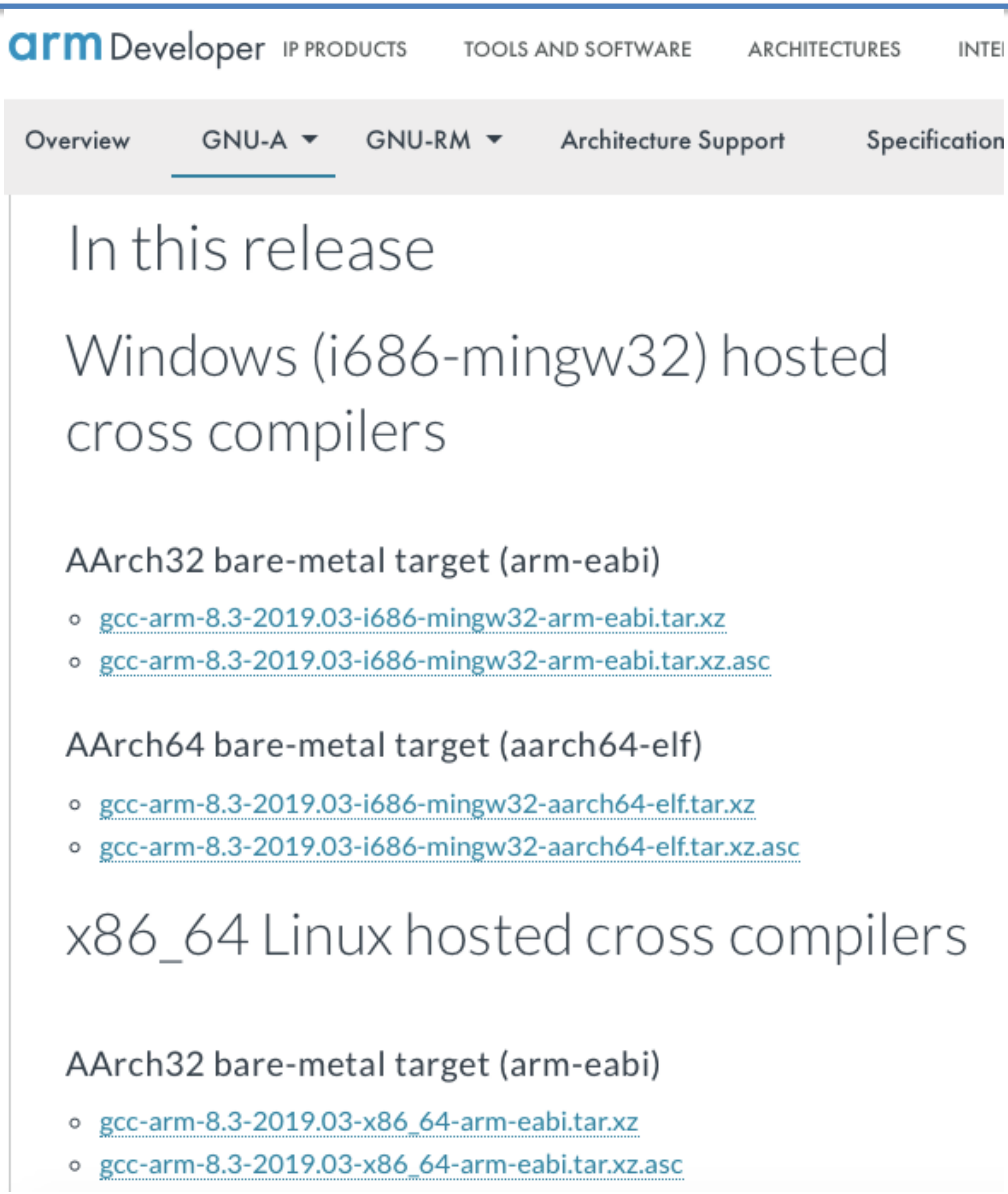
Released: March 29, 2019

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

ARM GNU-A

DR JEFF SOFTWARE
INDIE APP DEVELOPER

gcc



In this release

Windows (i686-mingw32) hosted cross compilers

AArch32 bare-metal target (arm-eabi)

- gcc-arm-8.3-2019.03-i686-mingw32-arm-eabi.tar.xz
- gcc-arm-8.3-2019.03-i686-mingw32-arm-eabi.tar.xz.asc

AArch64 bare-metal target (aarch64-elf)

- gcc-arm-8.3-2019.03-i686-mingw32-aarch64-elf.tar.xz
- gcc-arm-8.3-2019.03-i686-mingw32-aarch64-elf.tar.xz.asc

x86_64 Linux hosted cross compilers

AArch32 bare-metal target (arm-eabi)

- gcc-arm-8.3-2019.03-x86_64-arm-eabi.tar.xz
- gcc-arm-8.3-2019.03-x86_64-arm-eabi.tar.xz.asc

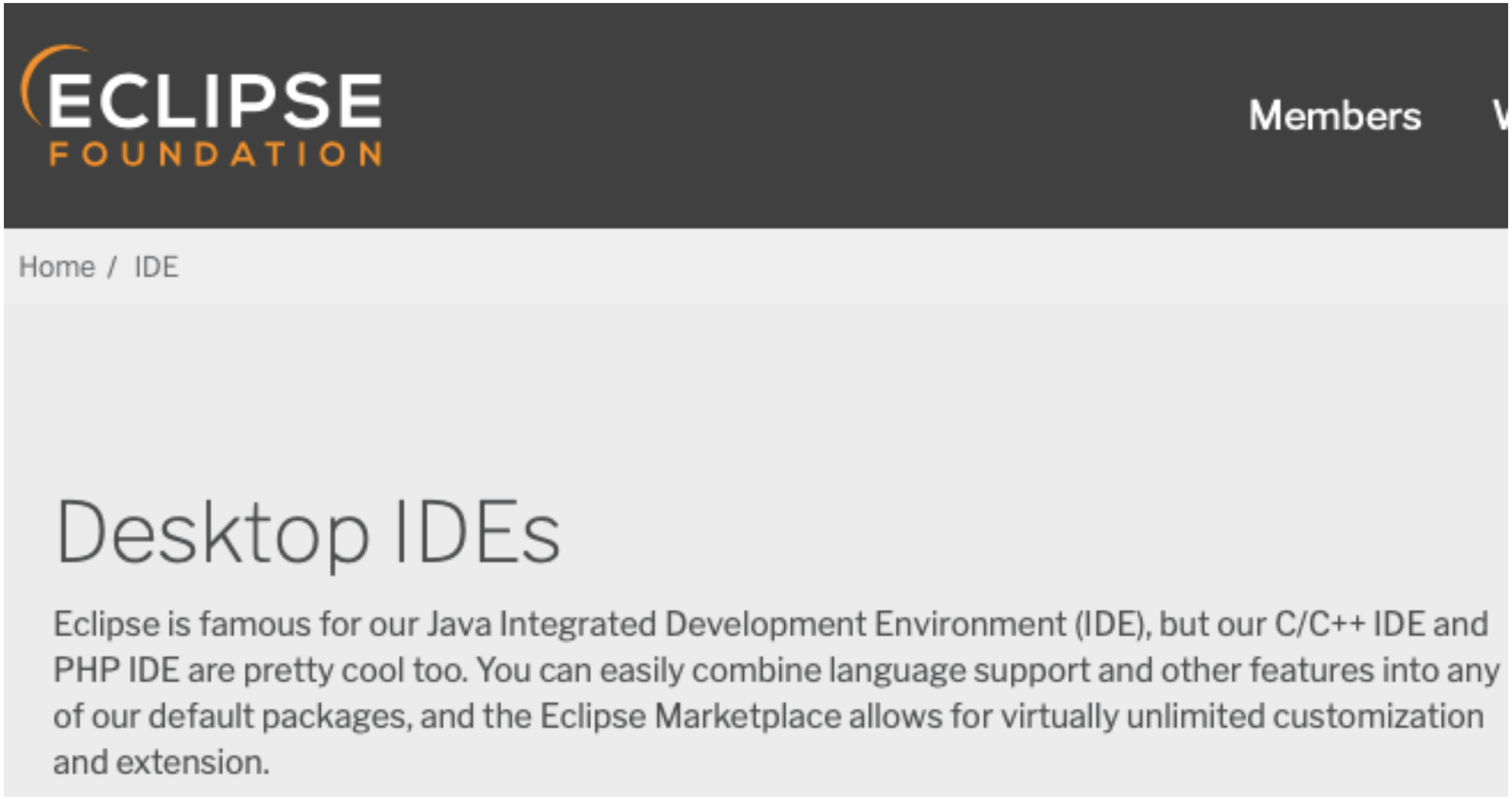
CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

Lab

Eclipse

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

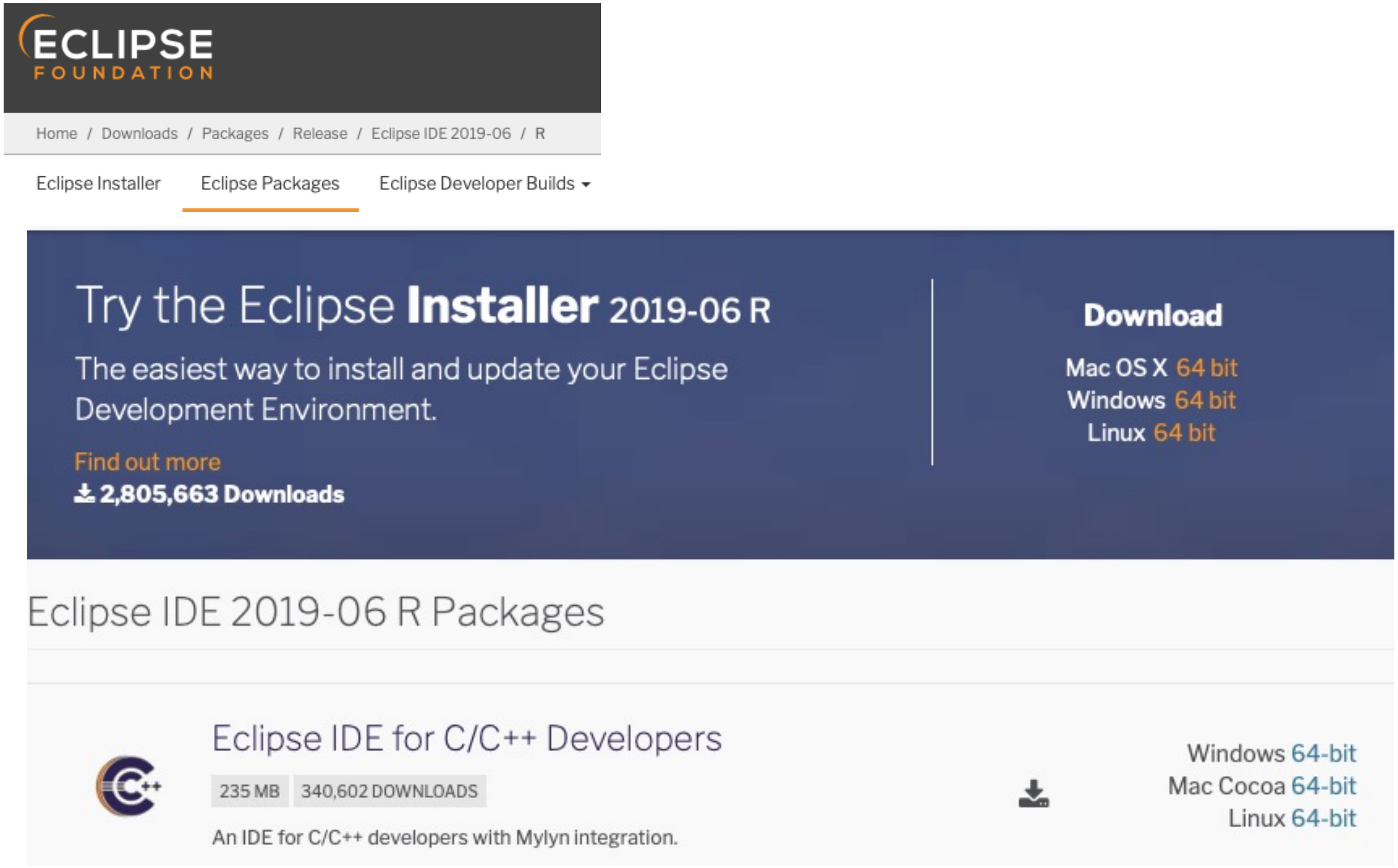
Eclipse



Desktop IDEs

Eclipse is famous for our Java Integrated Development Environment (IDE), but our C/C++ IDE and PHP IDE are pretty cool too. You can easily combine language support and other features into any of our default packages, and the Eclipse Marketplace allows for virtually unlimited customization and extension.

Eclipse



ECLIPSE FOUNDATION

Home / Downloads / Packages / Release / Eclipse IDE 2019-06 / R

Eclipse Installer | Eclipse Packages | Eclipse Developer Builds

Try the Eclipse **Installer** 2019-06 R

The easiest way to install and update your Eclipse Development Environment.

Find out more

2,805,663 Downloads

Download

Mac OS X 64 bit
Windows 64 bit
Linux 64 bit

Eclipse IDE 2019-06 R Packages

Eclipse IDE for C/C++ Developers

235 MB | 340,602 DOWNLOADS

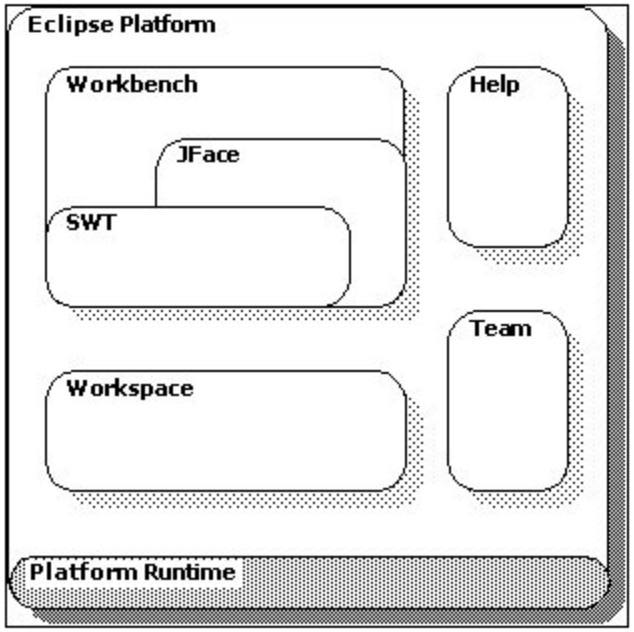
An IDE for C/C++ developers with Mylyn integration.

Windows 64-bit
Mac Cocoa 64-bit
Linux 64-bit

Eclipse Platform

Eclipse platform overview

The Eclipse platform itself is structured as subsystems which are implemented in one or more plug-ins. The subsystems are built on top of a small runtime engine. The figure below depicts a simplified view.



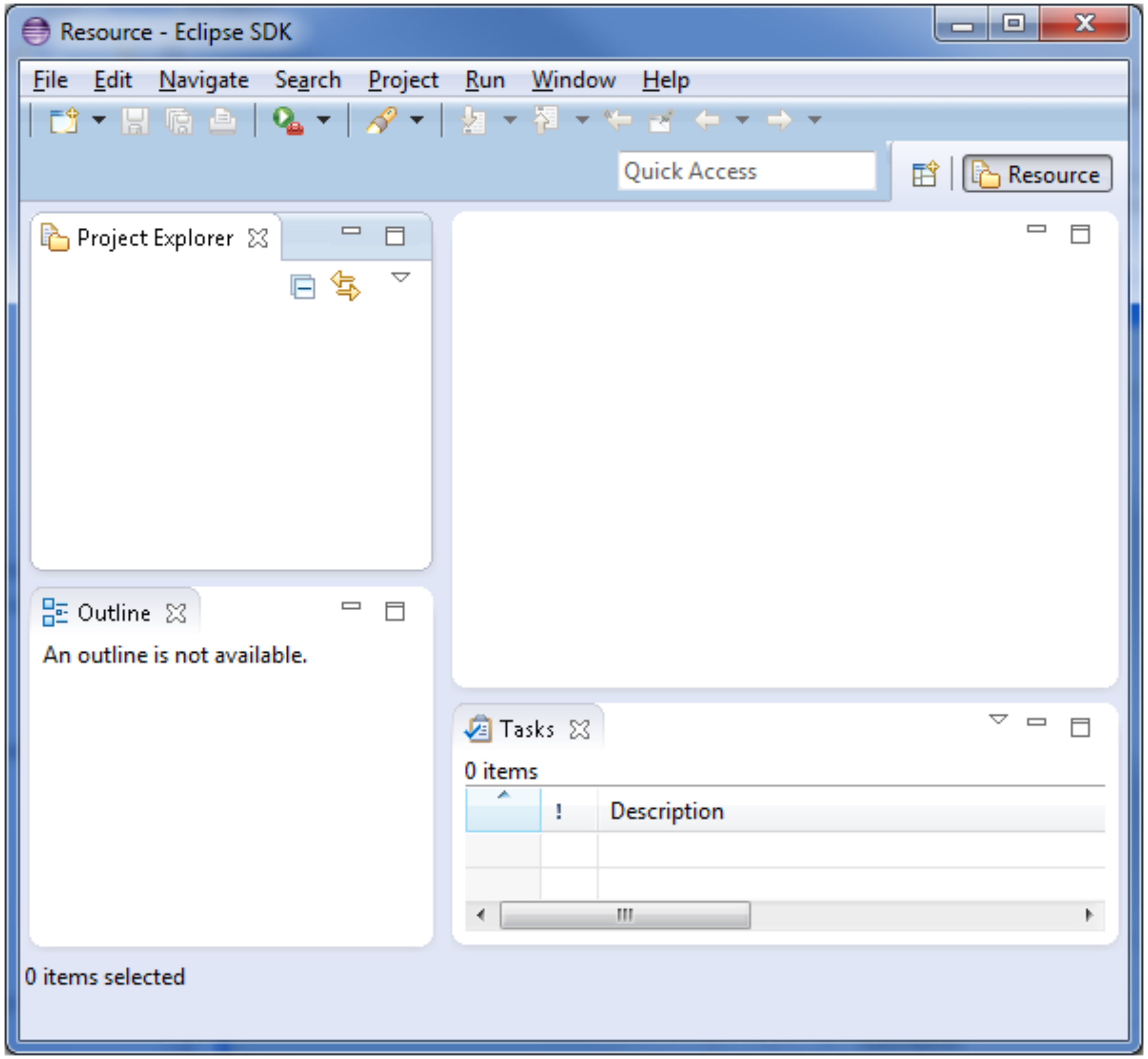
Workbench

The term *Workbench* refers to the desktop development environment. The Workbench aims to achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of workspace resources.

CSUN CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

Eclipse SDK

DR JEFF SOFTWARE INDIE APP DEVELOPER
© Jeff Drobman 2016-2023



An outline is not available.

0 items

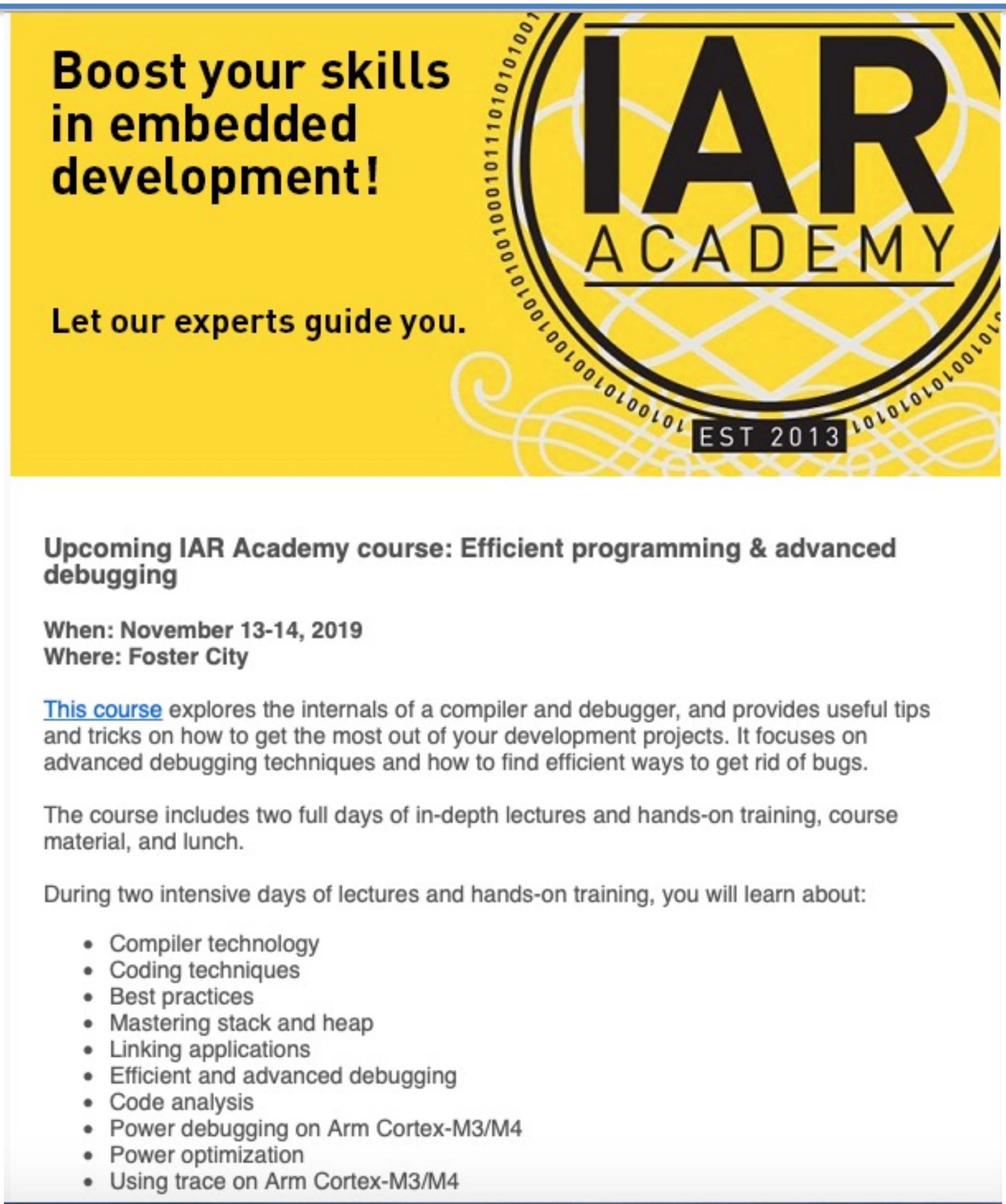
0 items selected

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP122

Lab

MISC IDE'S

IAR


Boost your skills in embedded development!
Let our experts guide you.
IAR ACADEMY
EST 2013

Upcoming IAR Academy course: Efficient programming & advanced debugging

When: November 13-14, 2019
Where: Foster City

This course explores the internals of a compiler and debugger, and provides useful tips and tricks on how to get the most out of your development projects. It focuses on advanced debugging techniques and how to find efficient ways to get rid of bugs.

The course includes two full days of in-depth lectures and hands-on training, course material, and lunch.

During two intensive days of lectures and hands-on training, you will learn about:

- Compiler technology
- Coding techniques
- Best practices
- Mastering stack and heap
- Linking applications
- Efficient and advanced debugging
- Code analysis
- Power debugging on Arm Cortex-M3/M4
- Power optimization
- Using trace on Arm Cortex-M3/M4