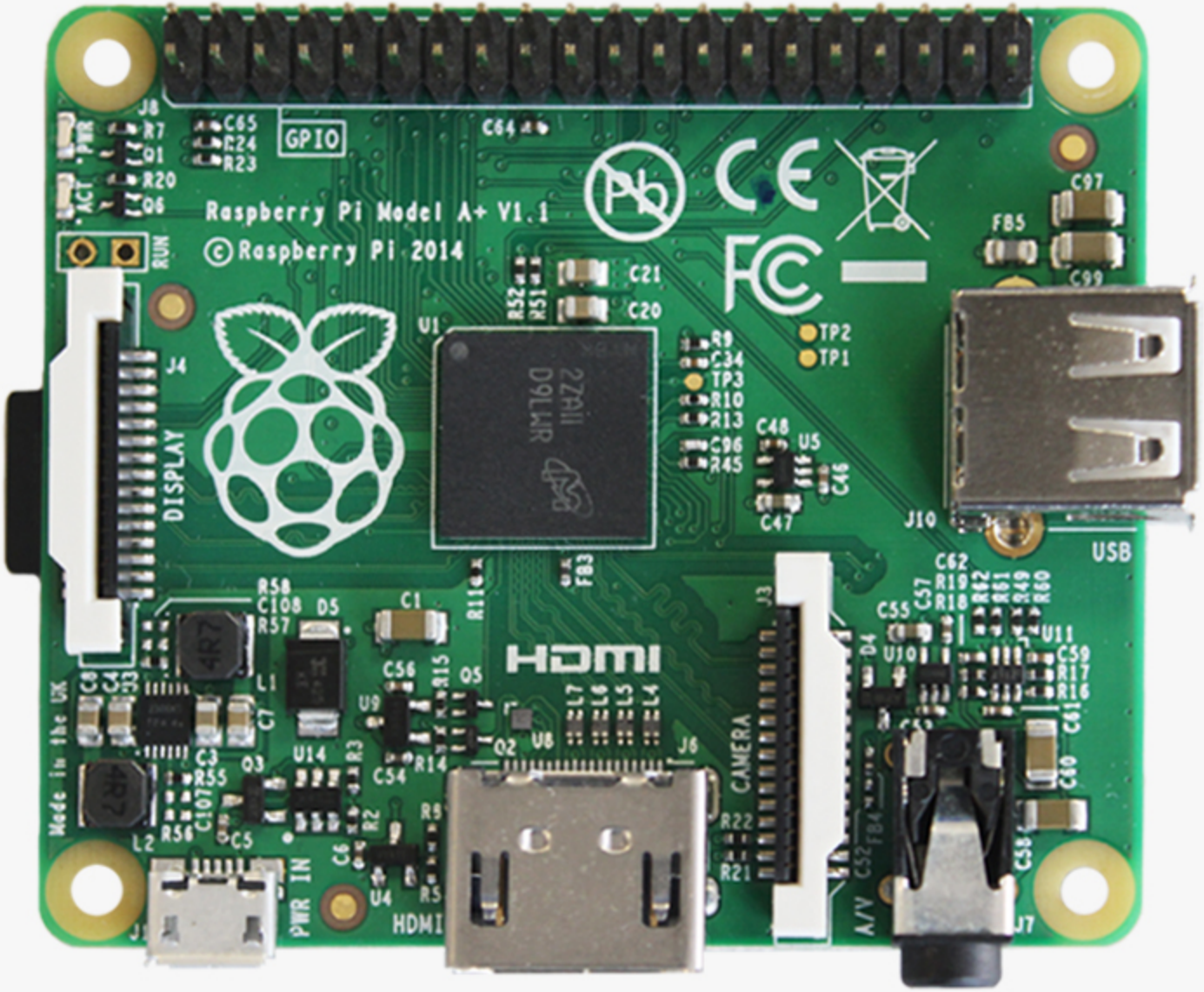

PRODUCTION
CS107e
Computer Systems from the Ground Up
SCENE
Wrap up
TAKE
2
DIRECTOR
Engler, Hanrahan, Zelenski, Benitez. Bui, Bush, Rizwan
CAMERA
DATE
Winter 2016

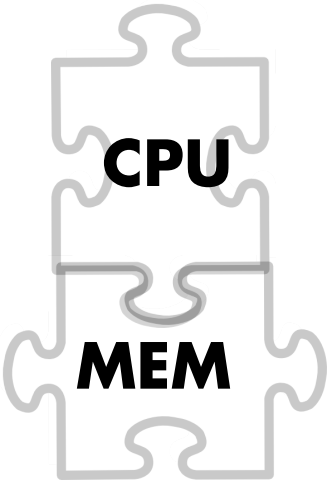
Goal 1

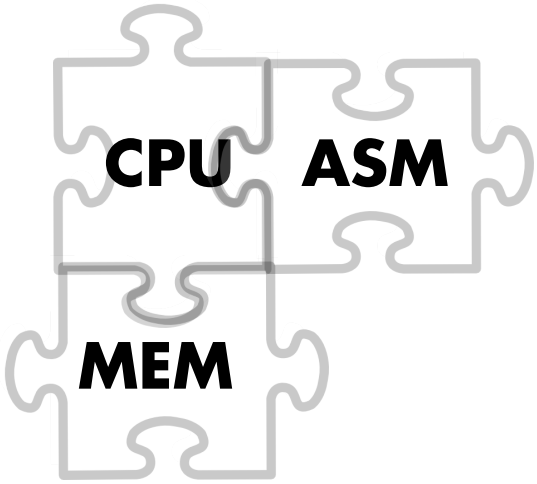
Understand How Computers Work

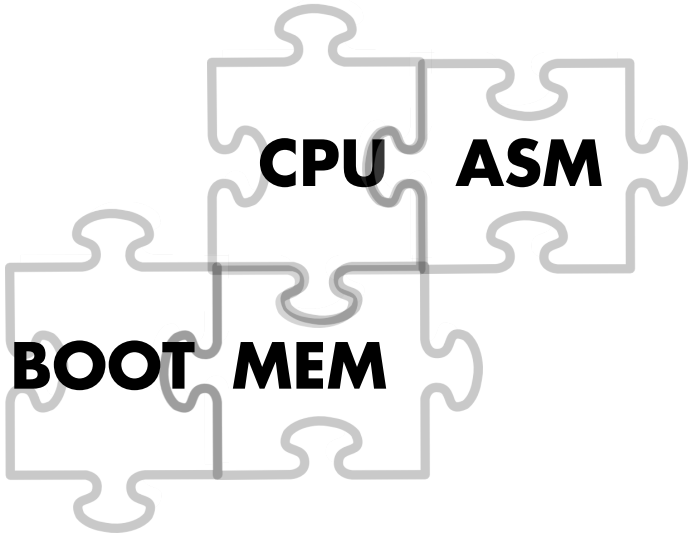
Computer Systems from the Ground Up

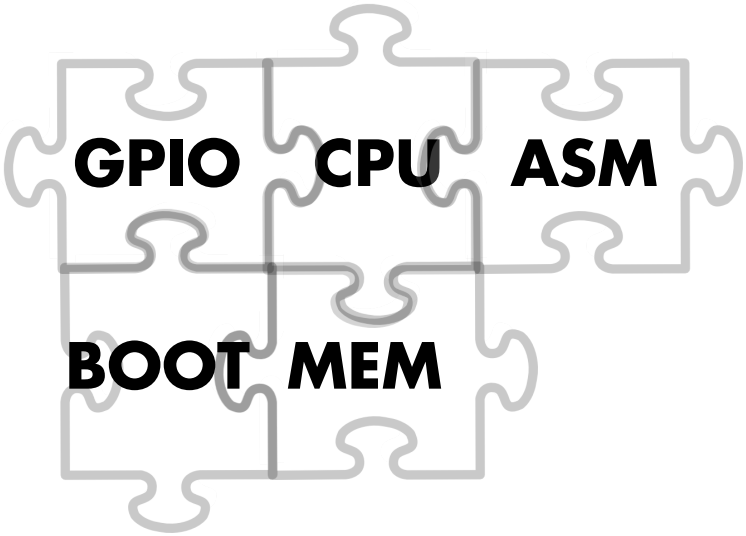


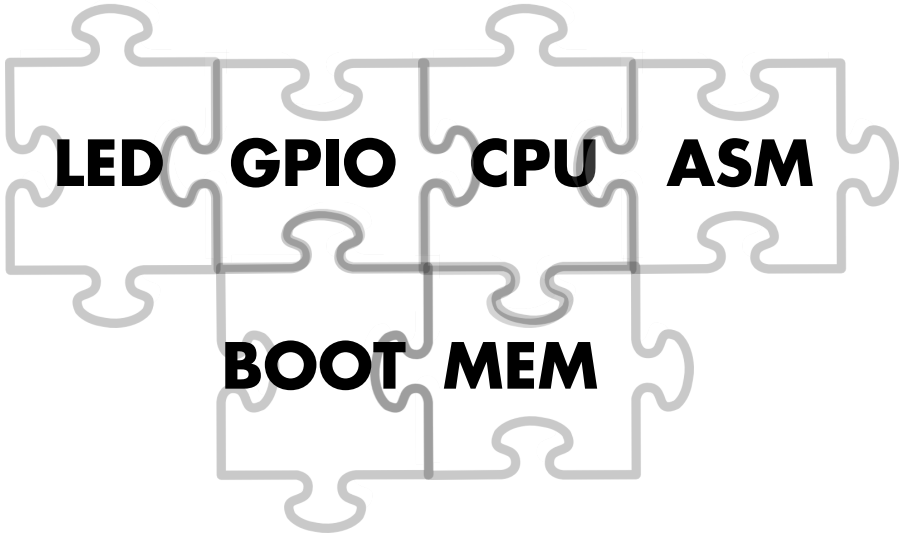
Bare-metal on the Raspberry Pi

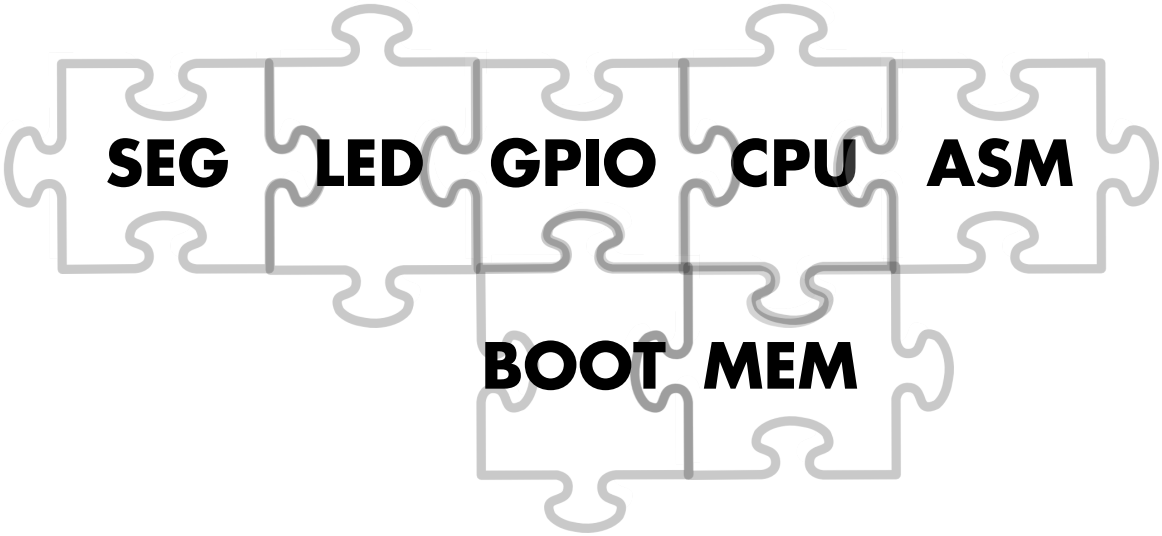

CPU
MEM

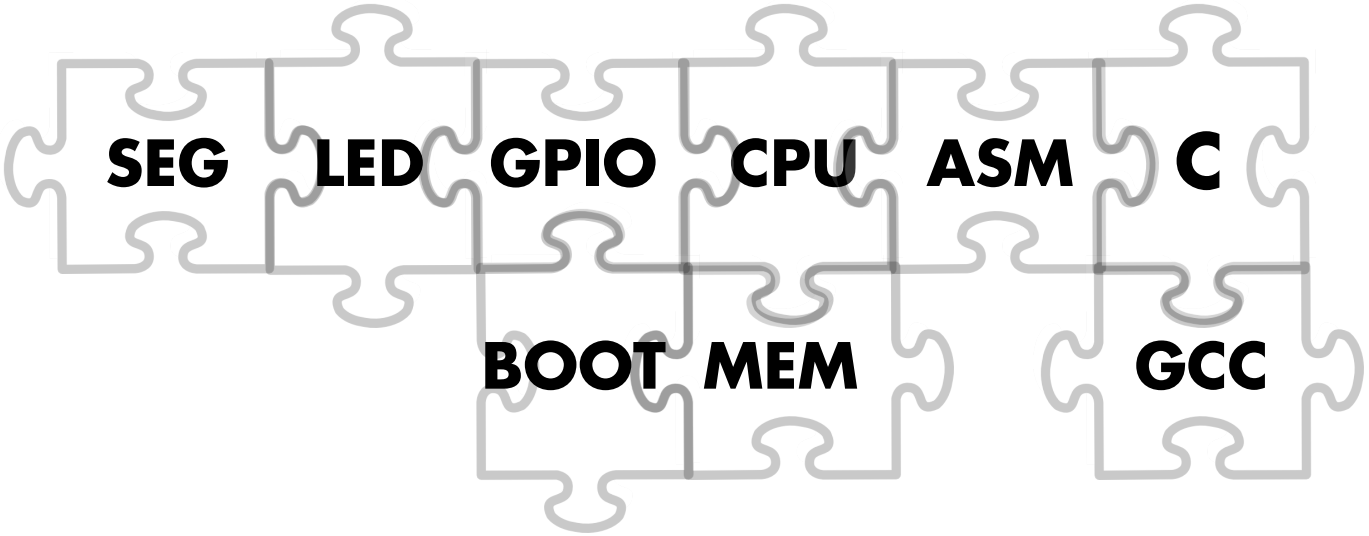

CPU
ASM
MEM

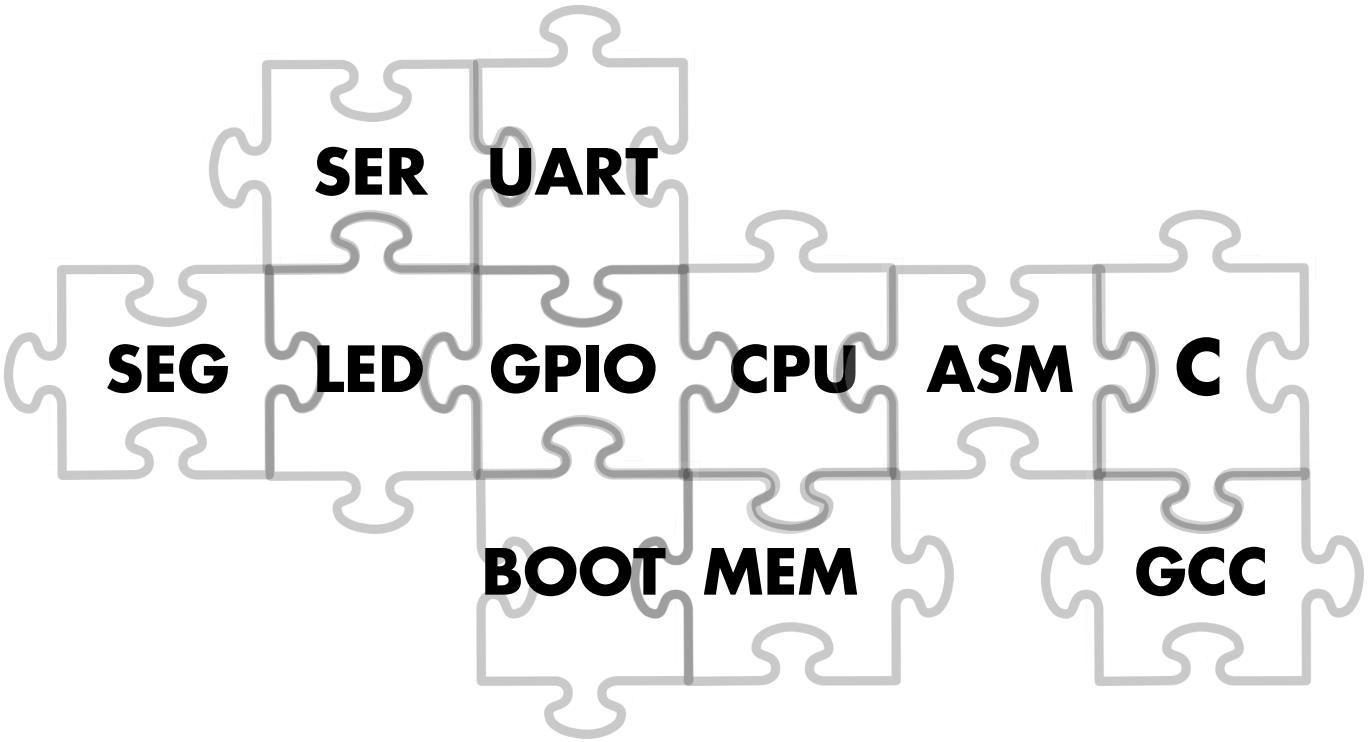

CPU
ASM
BOOT
MEM

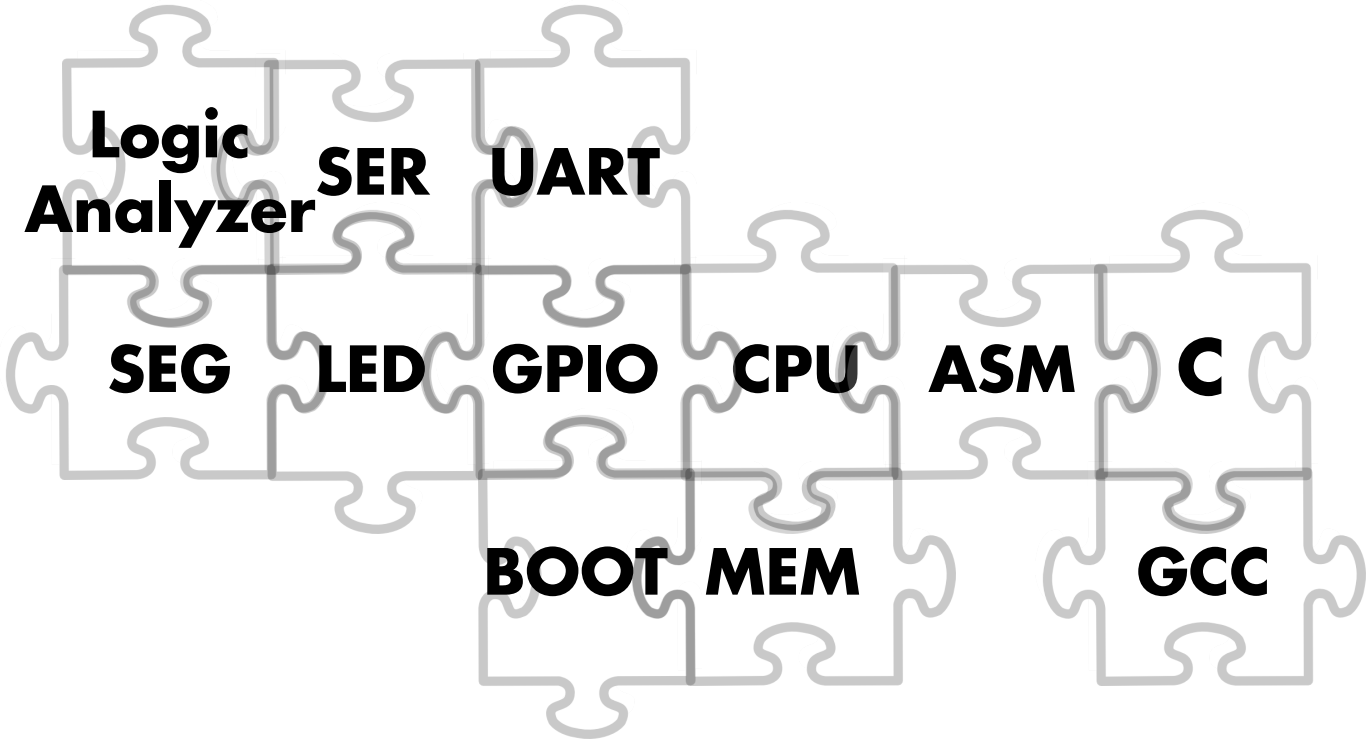

GPIO
CPU
ASM
BOOT
MEM

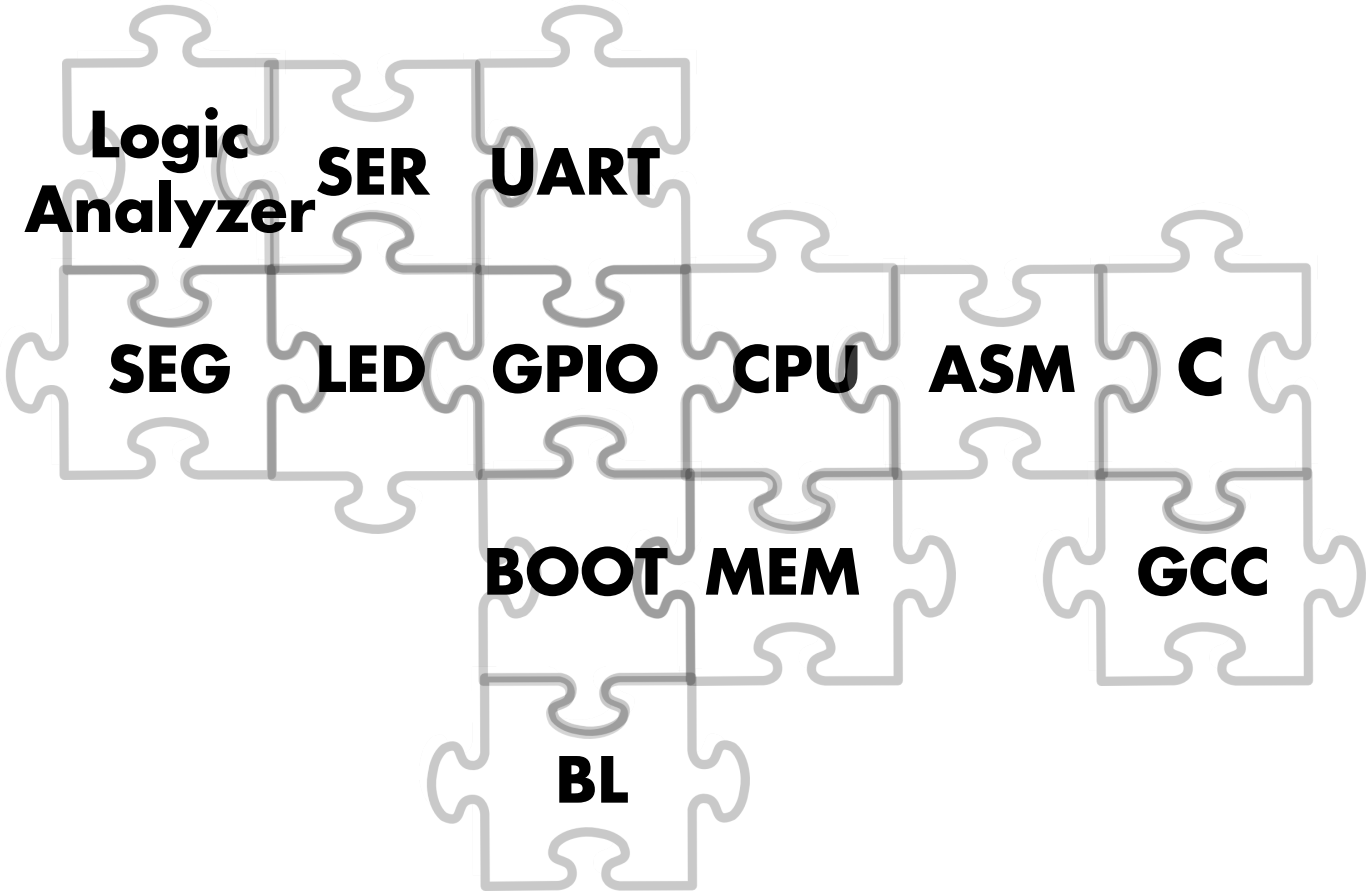

LED
GPIO
CPU
ASM
BOOT
MEM

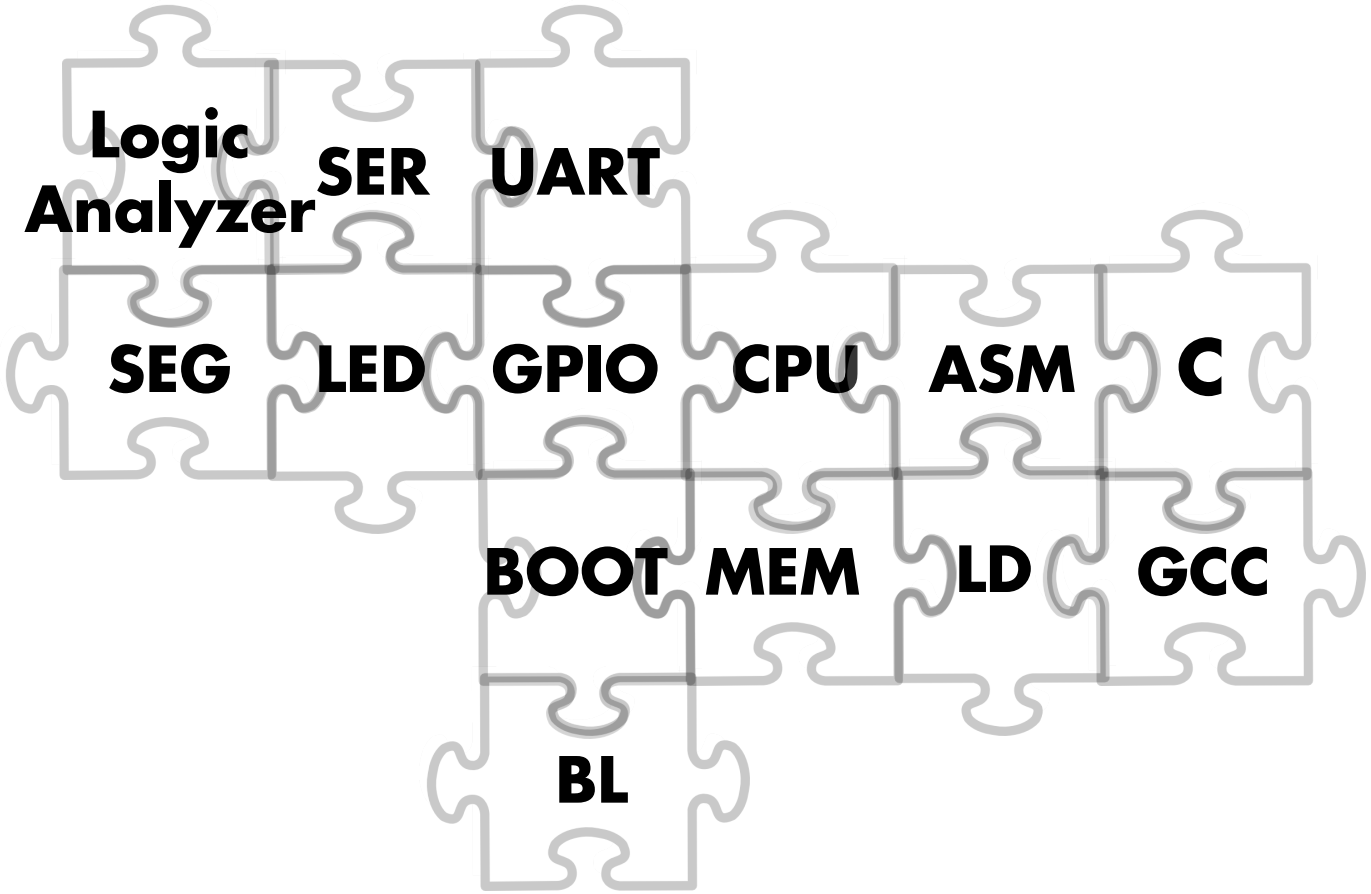

SEG
LED
GPIO
CPU
ASM
BOOT
MEM

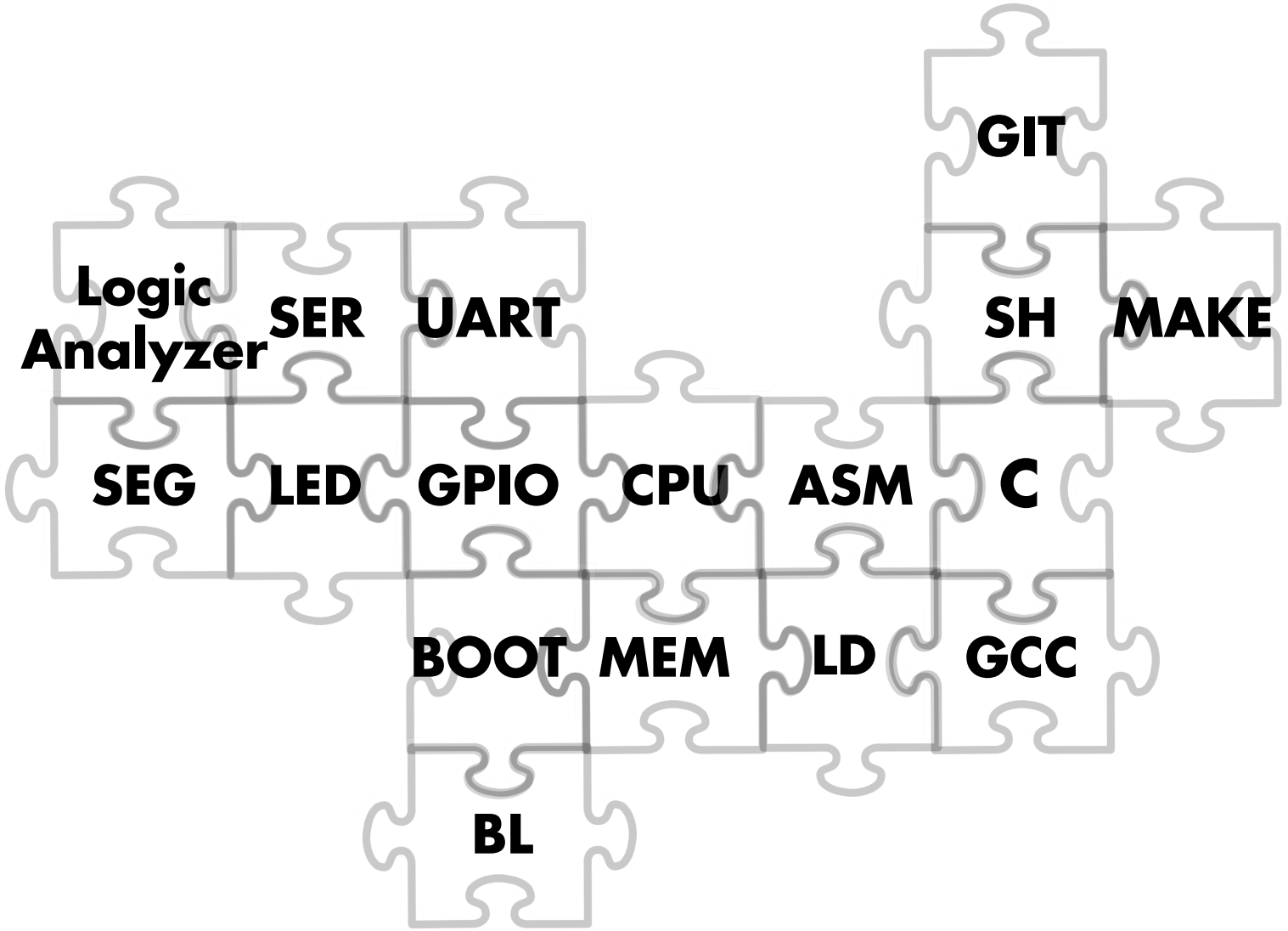

SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
GCC

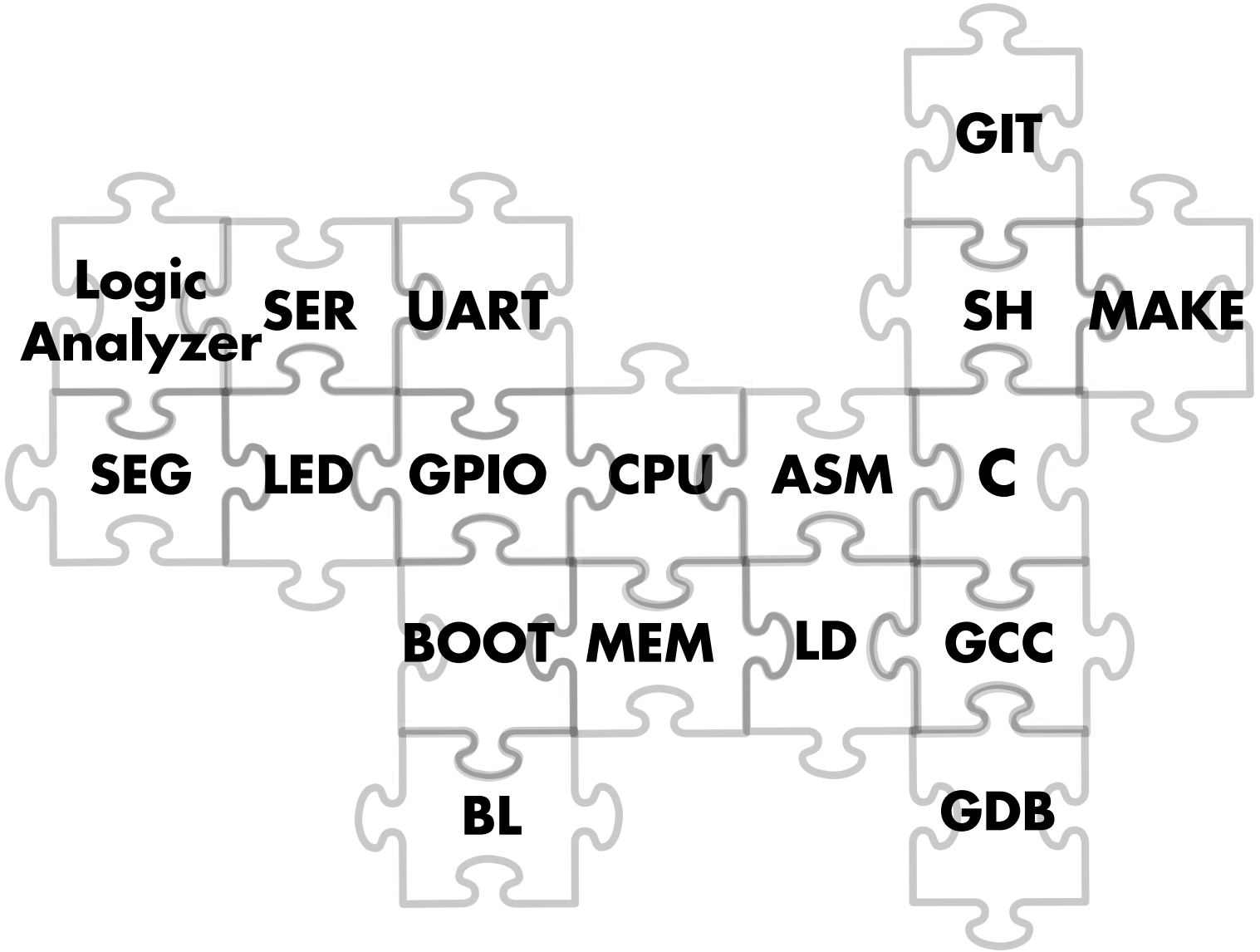

SER
UART
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
GCC

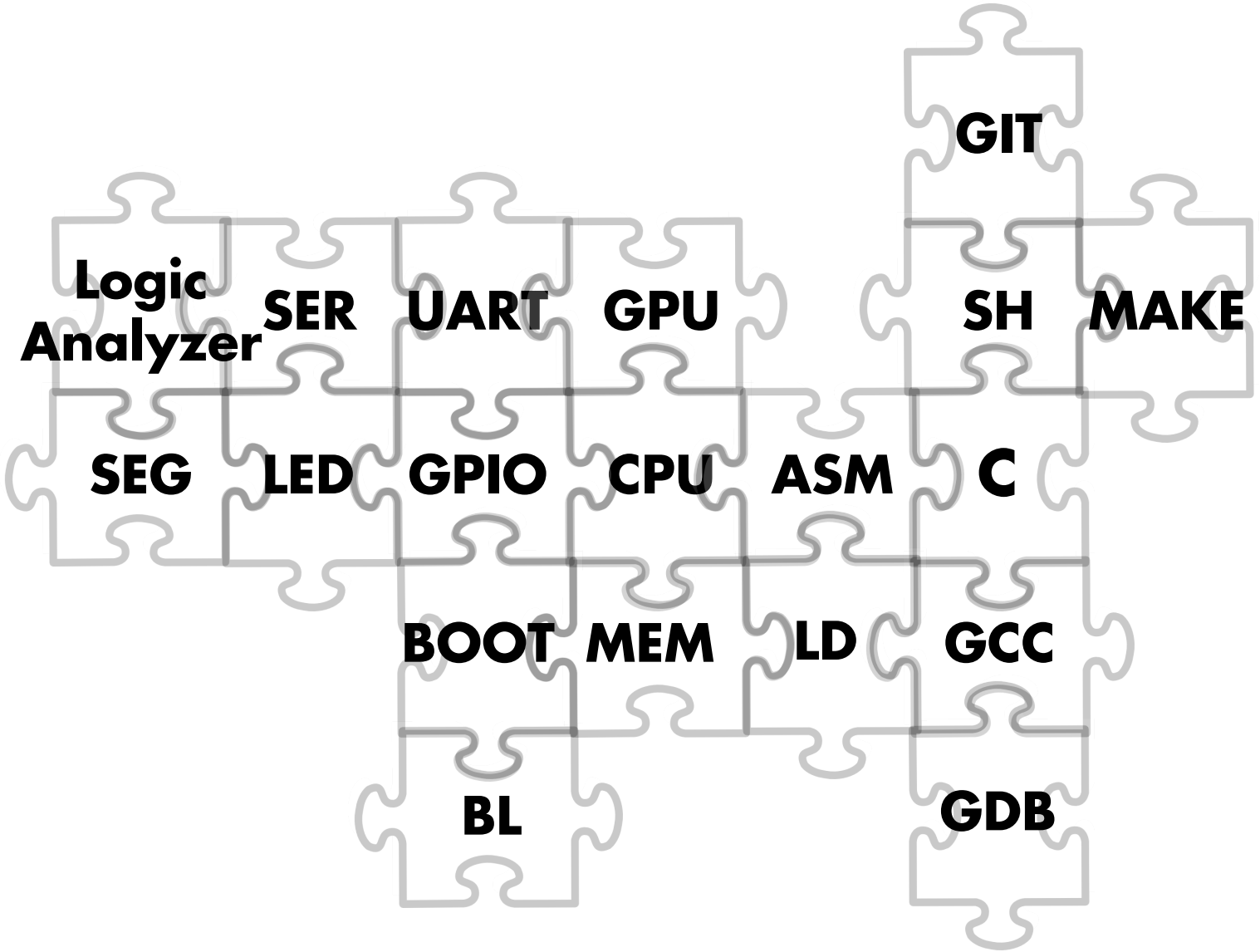

Logic Analyzer
SER
UART
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
GCC

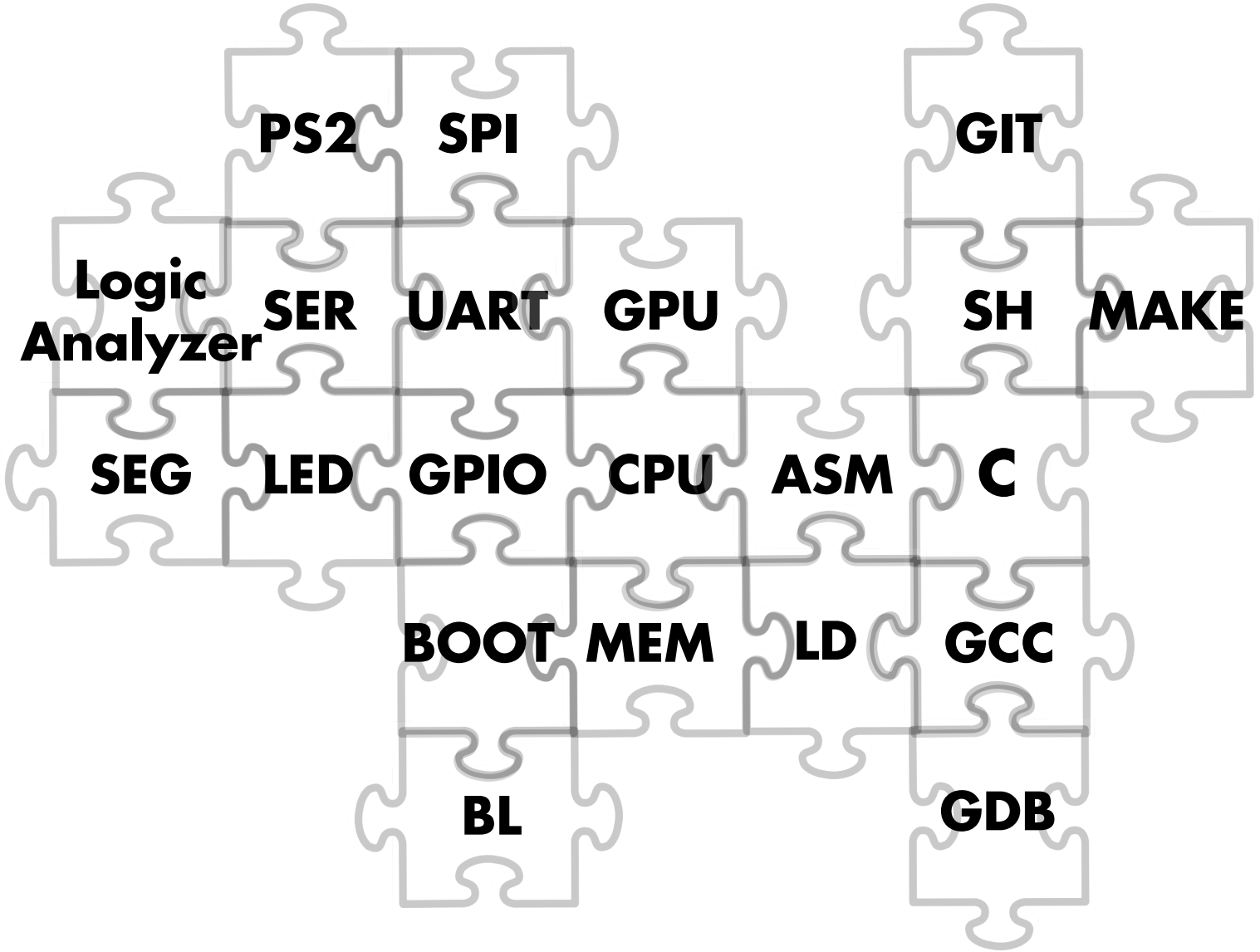

Logic Analyzer
SER
UART
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
GCC
BL

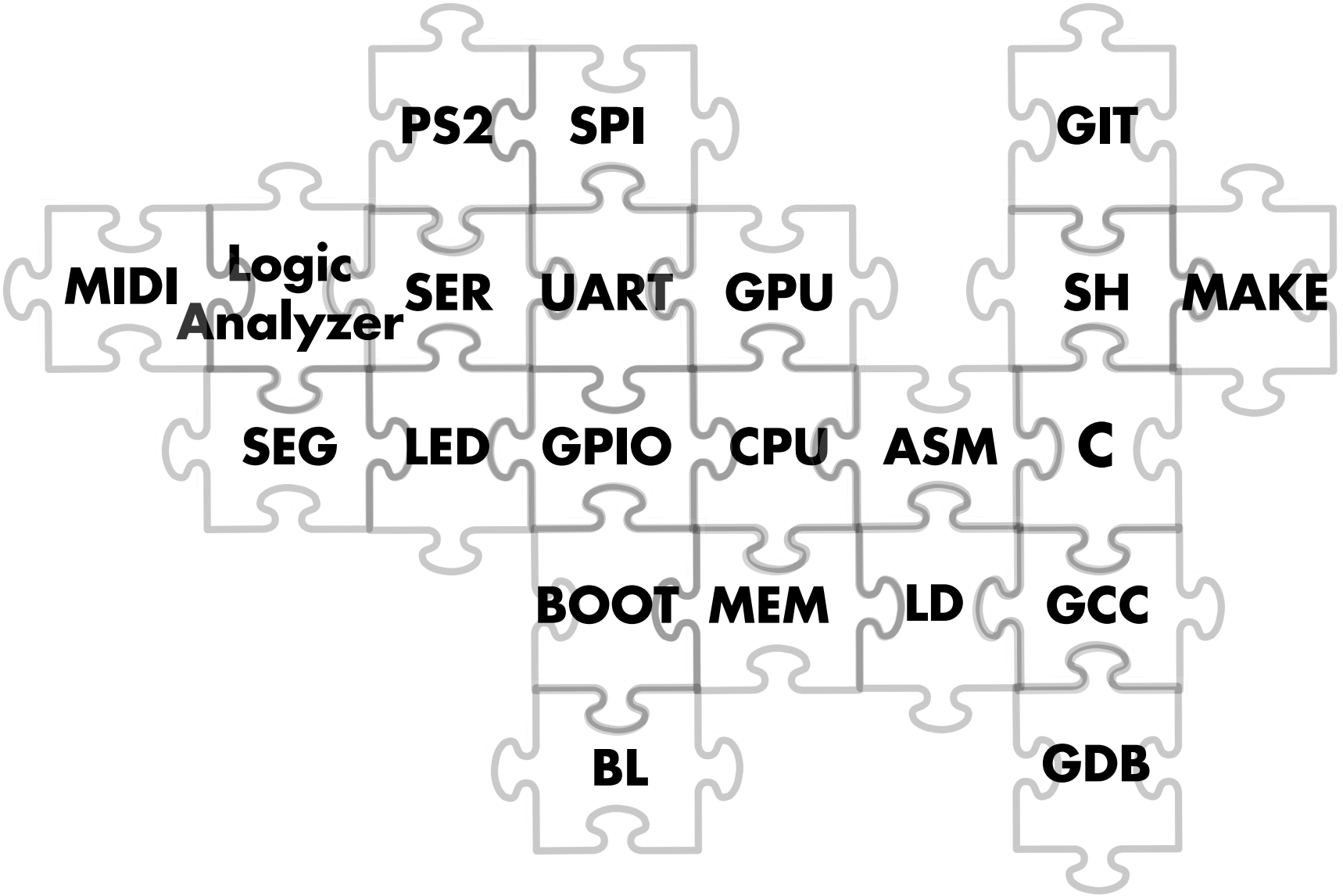

Logic Analyzer
SER
UART
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
LD
GCC
BL

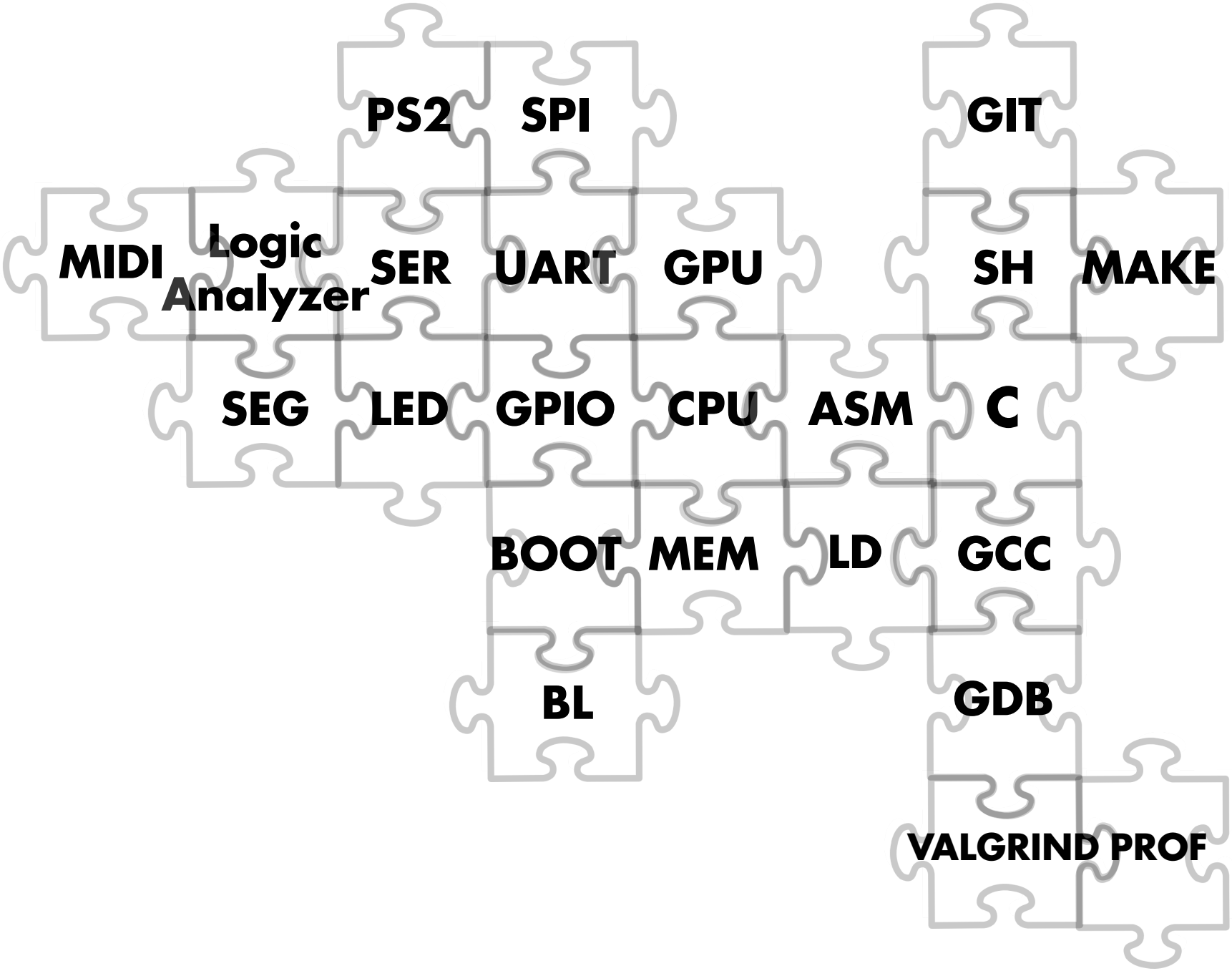

GIT
Logic Analyzer
SER
UART
SH
MAKE
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
LD
GCC
BL


GIT
Logic Analyzer
SER
UART
SH
MAKE
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
LD
GCC
BL
GDB

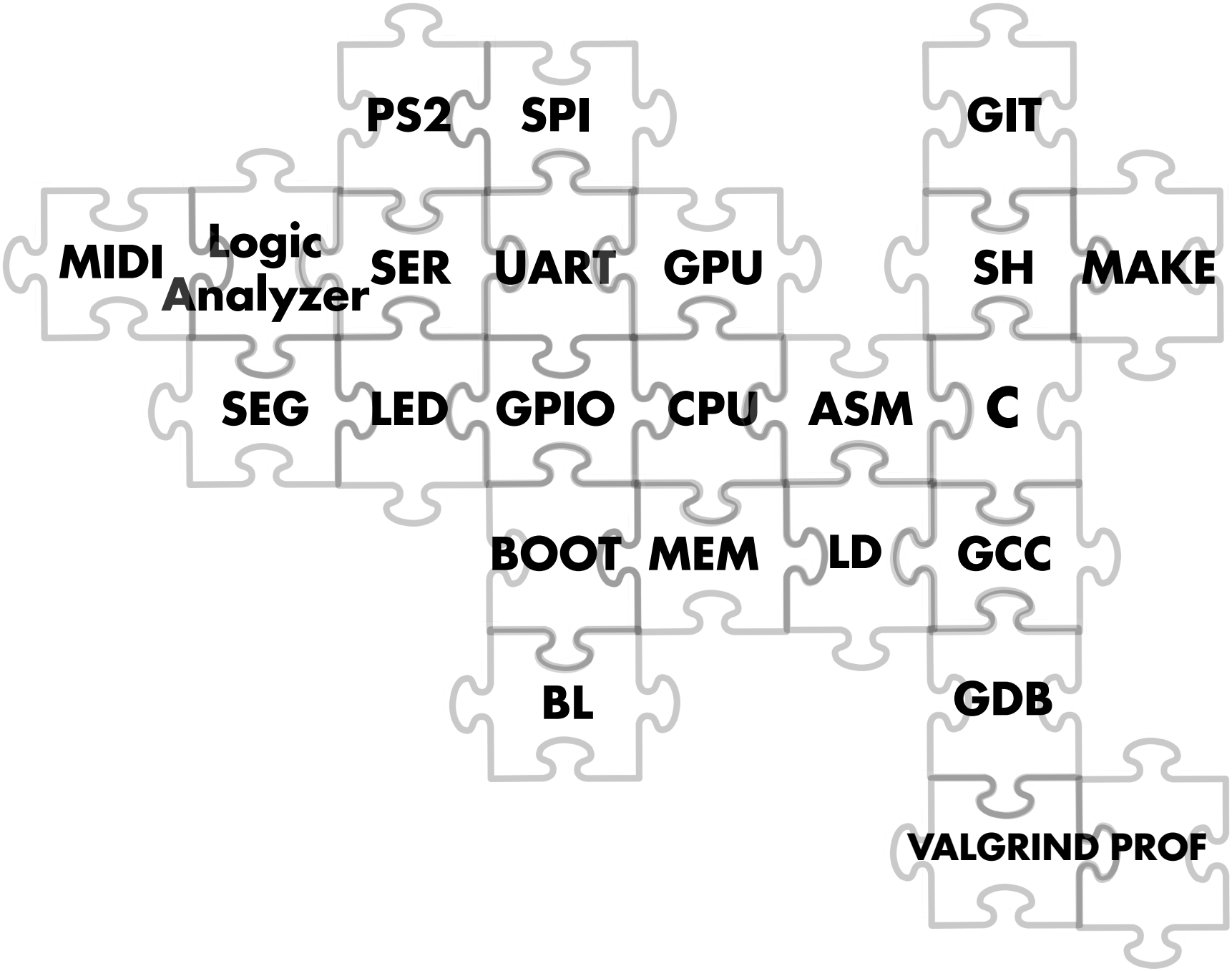

GIT
Logic Analyzer
SER
UART
GPU
SH
MAKE
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
LD
GCC
BL
GDB

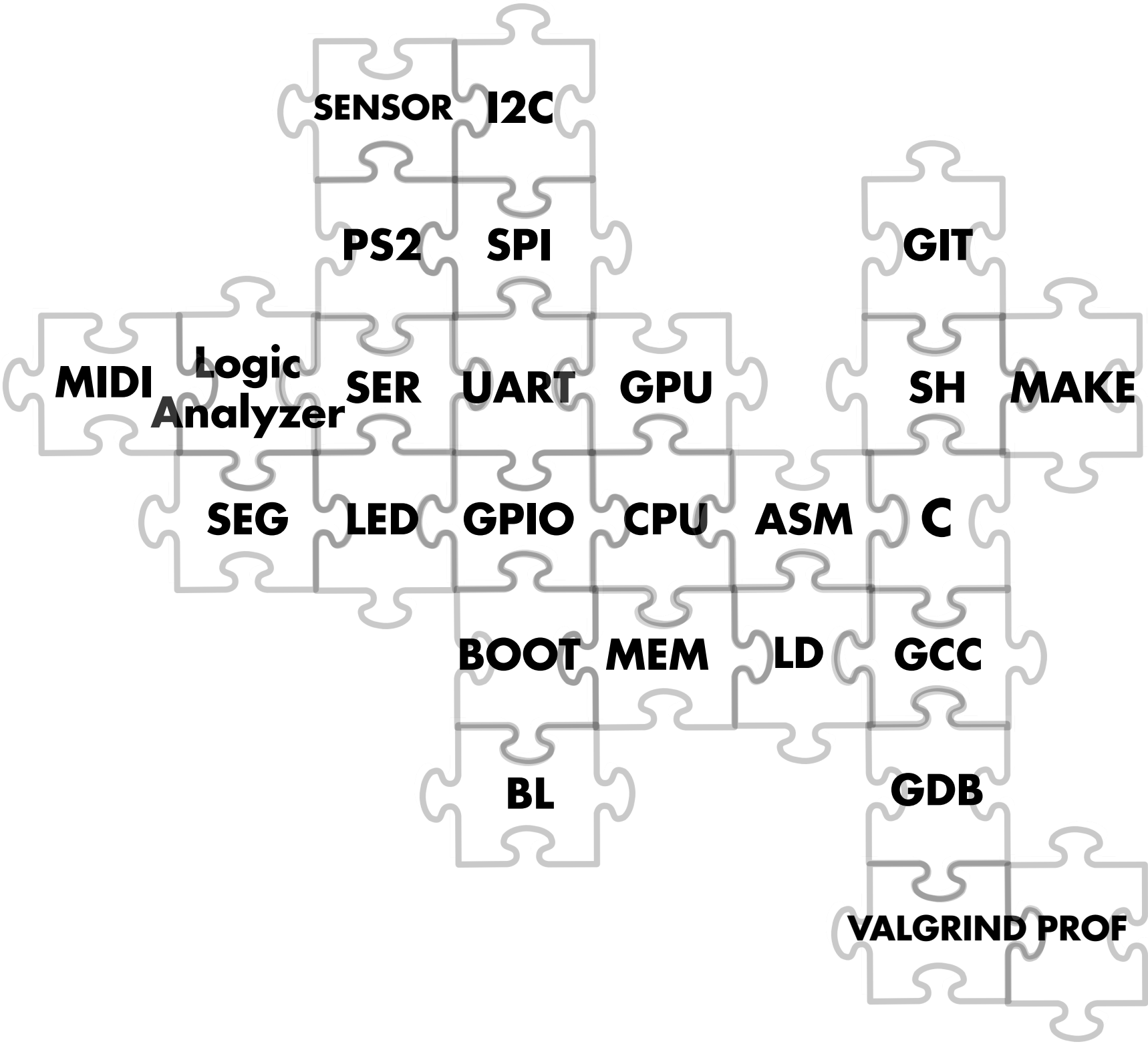

PS2
SPI
GIT
Logic Analyzer
SER
UART
GPU
SH
MAKE
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
LD
GCC
BL
GDB

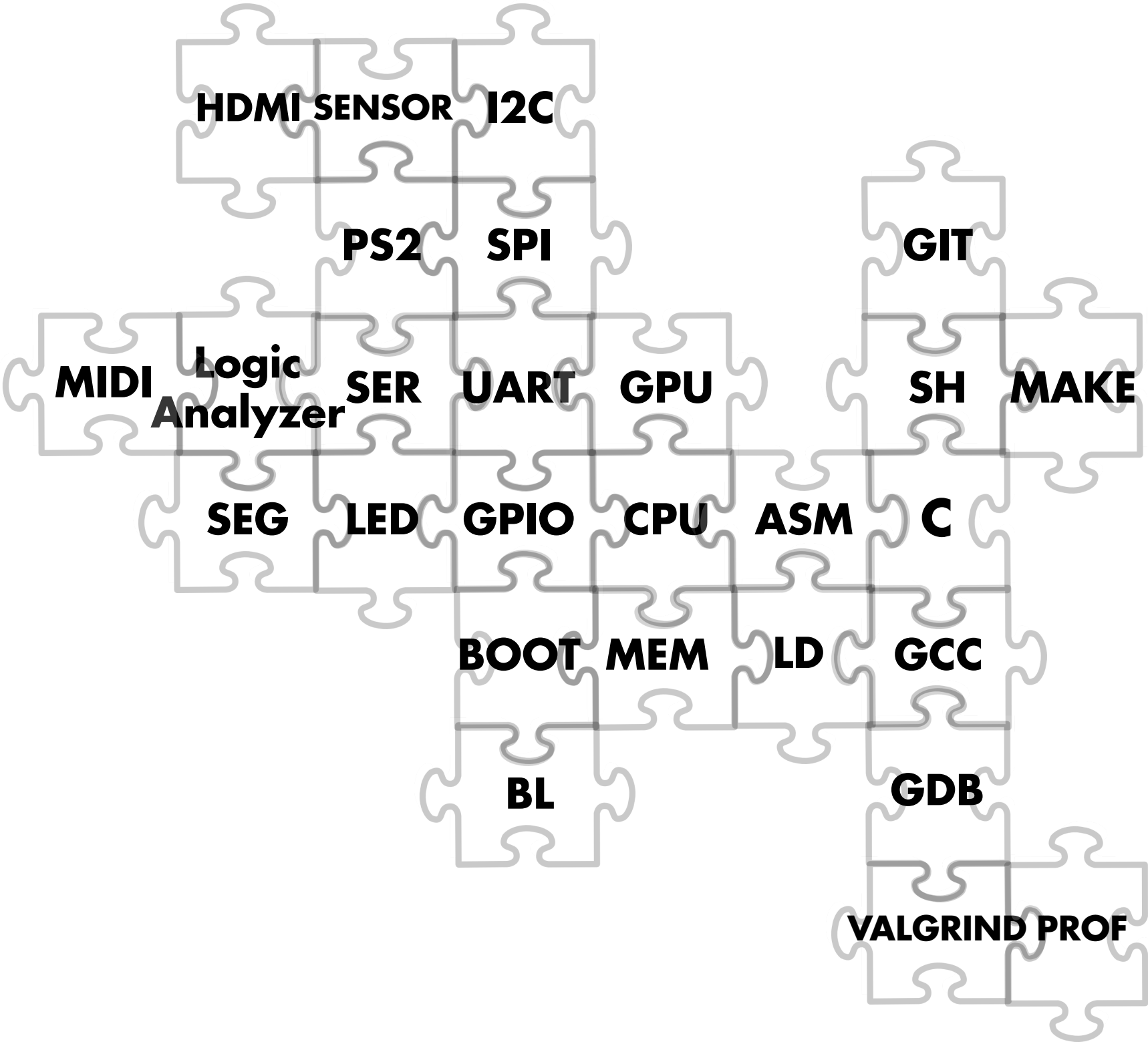

PS2
SPI
GIT
MIDI
Logic Analyzer
SER
UART
GPU
SH
MAKE
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
LD
GCC
BL
GDB

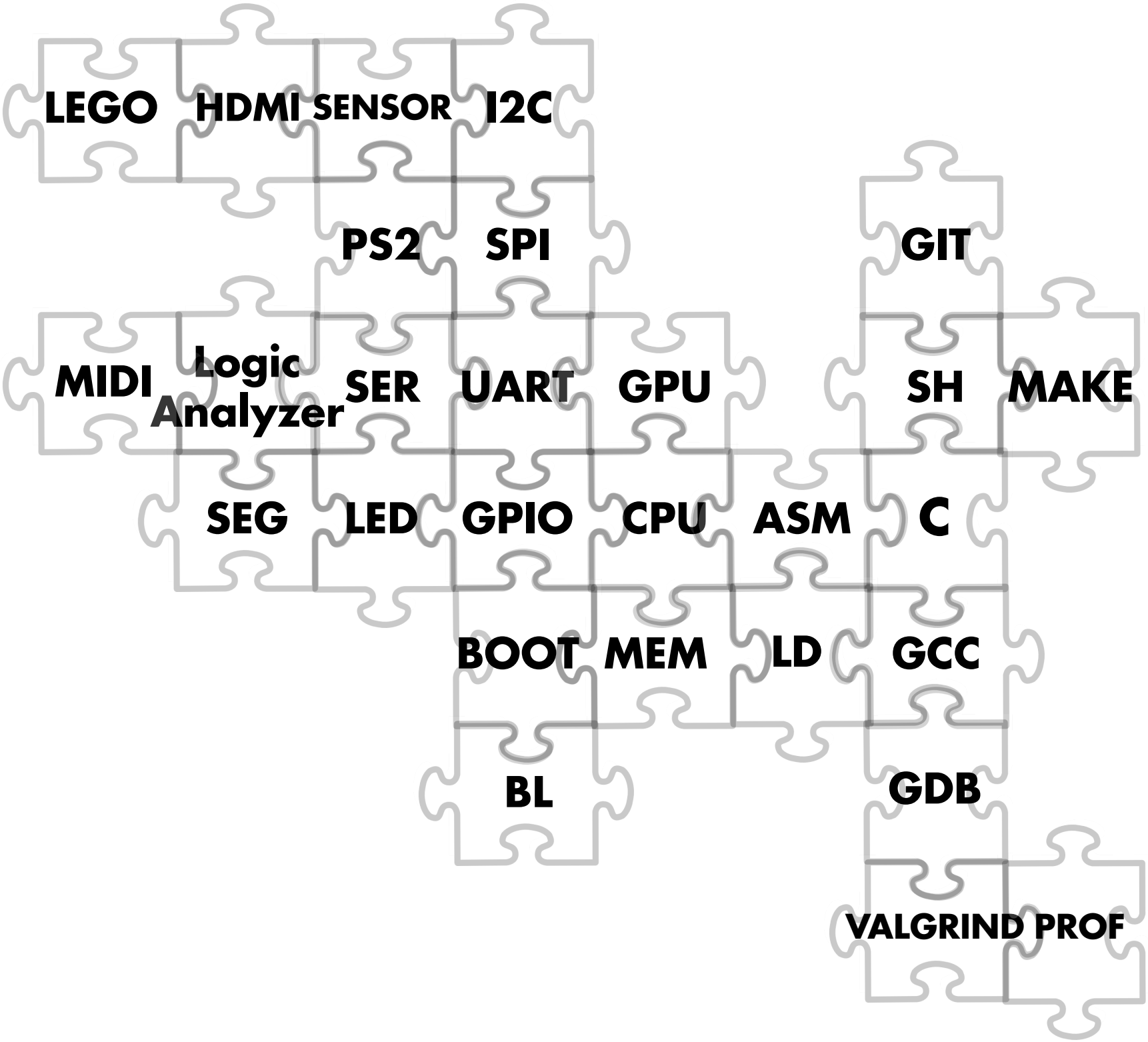

PS2
SPI
GIT
MIDI
Logic Analyzer
SER
UART
GPU
SH
MAKE
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
LD
GCC
BL
GDB
VALGRIND
PROF

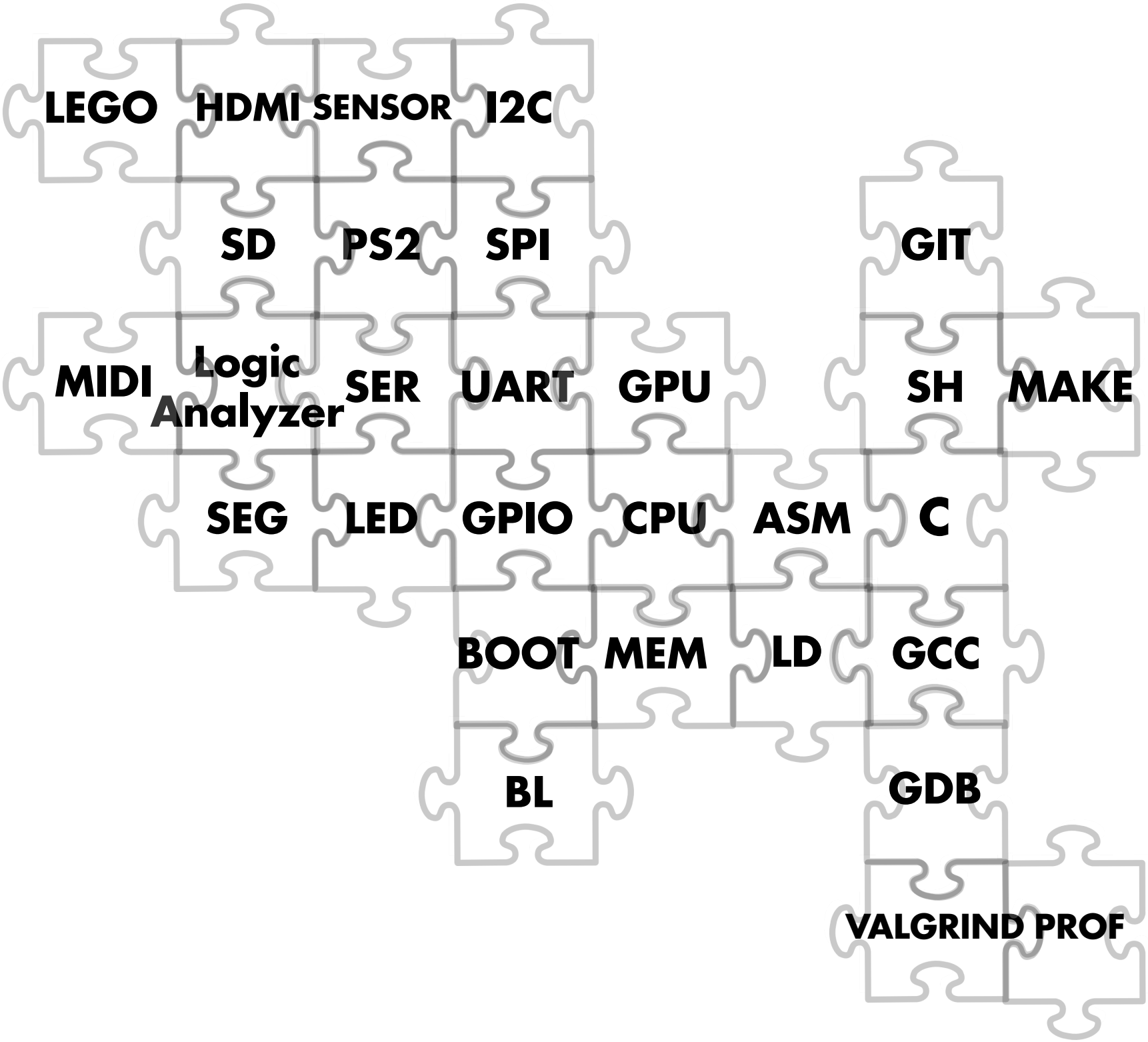
But It Keeps Growing!

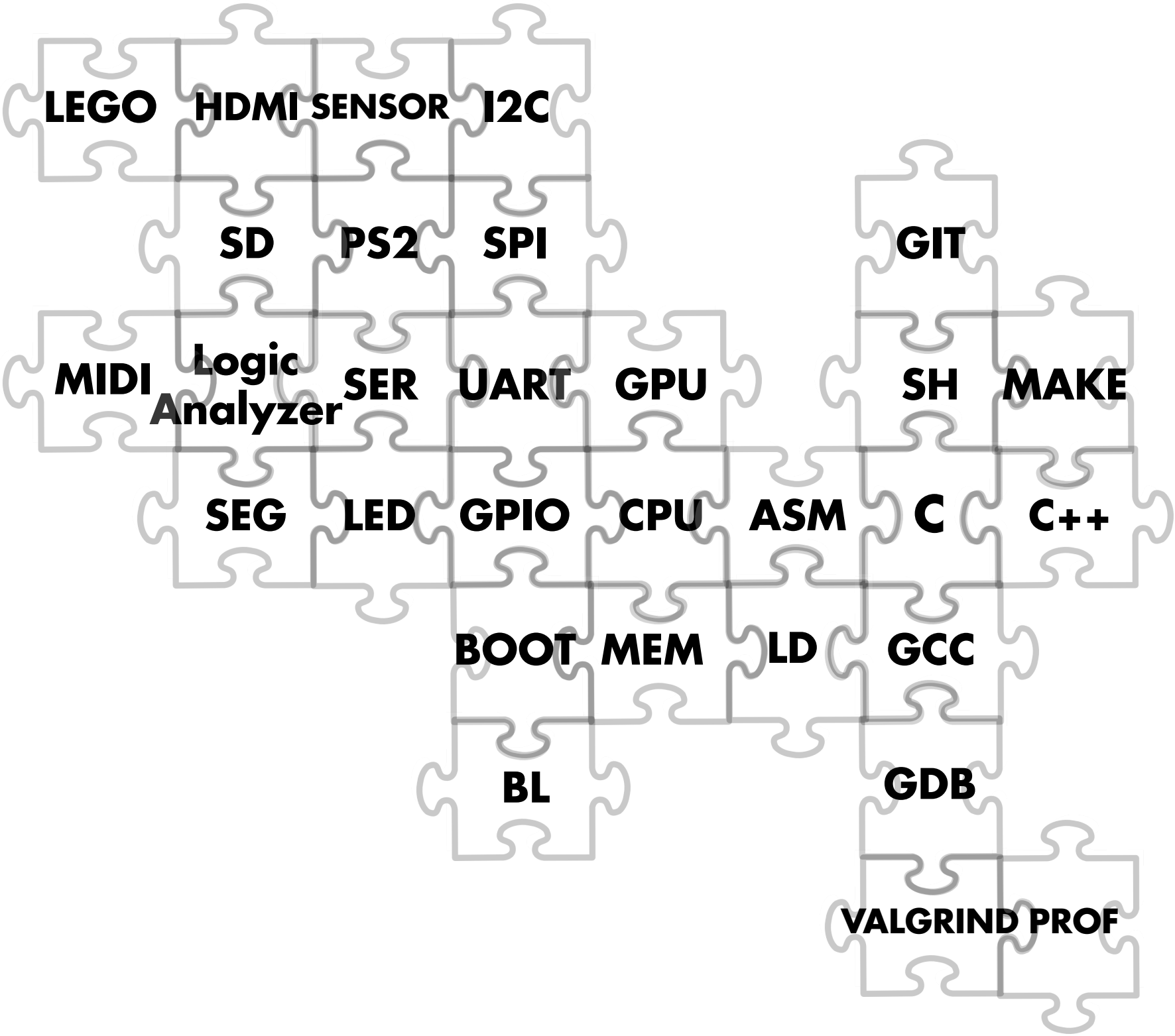

PS2
SPI
GIT
MIDI
Logic Analyzer
SER
UART
GPU
SH
MAKE
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
LD
GCC
BL
GDB
VALGRIND
PROF

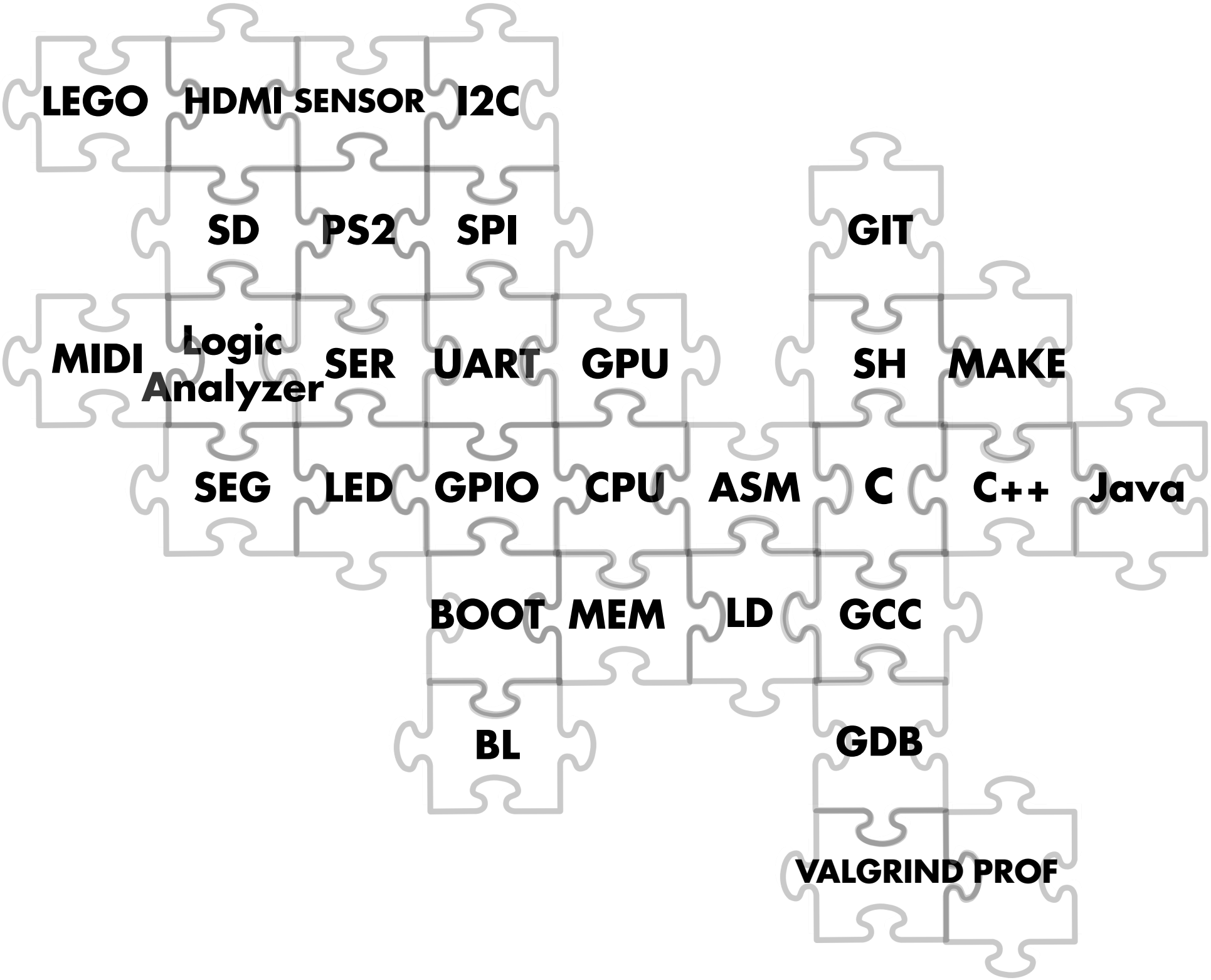

SENSOR
I2C
PS2
SPI
GIT
MIDI
Logic Analyzer
SER
UART
GPU
SH
MAKE
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
LD
GCC
BL
GDB
VALGRIND
PROF


HDMI
SENSOR
I2C
PS2
SPI
GIT
MIDI
Logic Analyzer
SER
UART
GPU
SH
MAKE
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
LD
GCC
BL
GDB
VALGRIND
PROF


LEGO
HDMI
SENSOR
I2C
PS2
SPI
GIT
MIDI
Logic Analyzer
SER
UART
GPU
SH
MAKE
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
LD
GCC
BL
GDB
VALGRIND
PROF

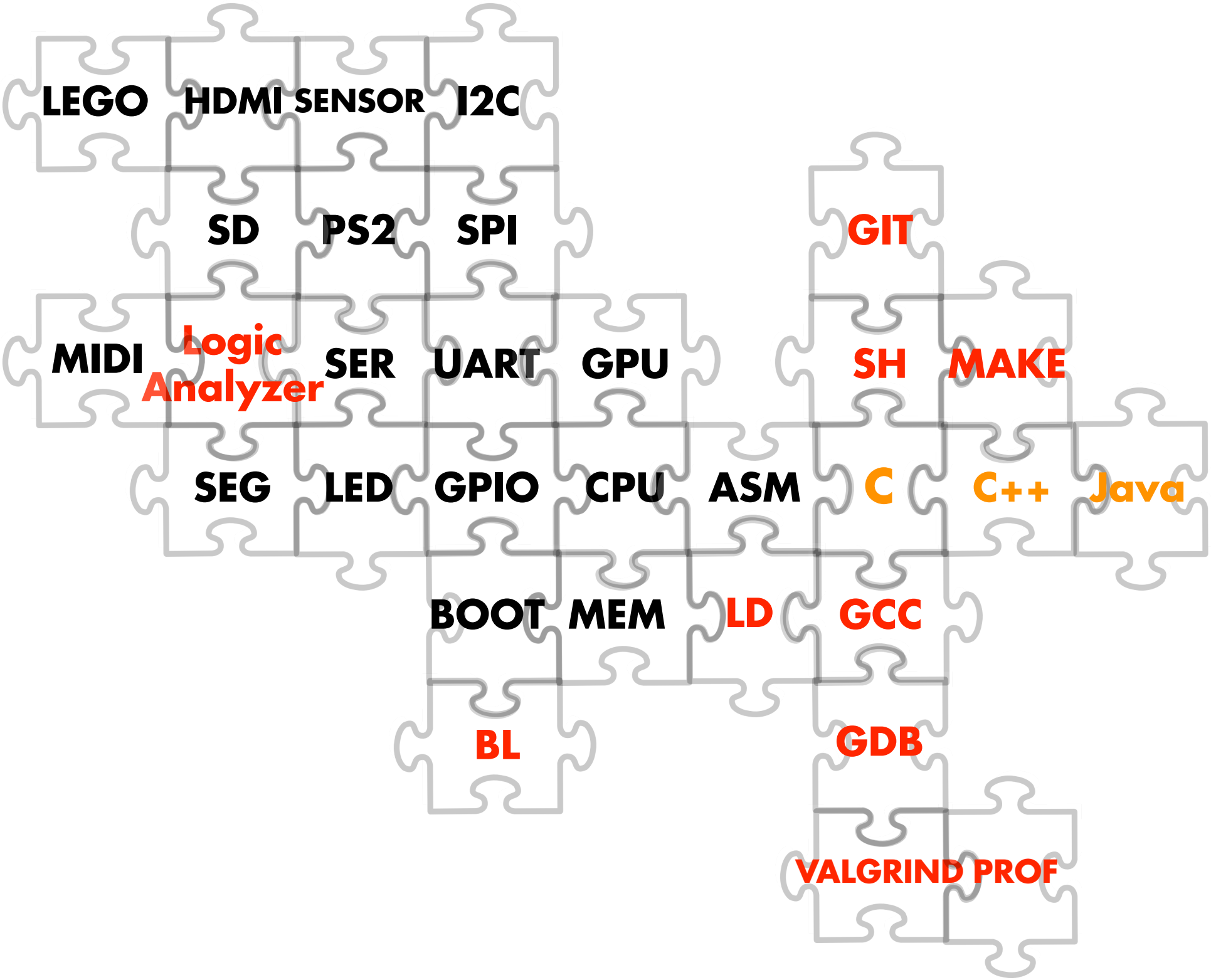

LEGO
HDMI
SENSOR
I2C
SD
PS2
SPI
GIT
MIDI
Logic Analyzer
SER
UART
GPU
SH
MAKE
SEG
LED
GPIO
CPU
ASM
C
BOOT
MEM
LD
GCC
BL
GDB
VALGRIND
PROF


LEGO
HDMI
SENSOR
I2C
SD
PS2
SPI
GIT
MIDI
Logic Analyzer
SER
UART
GPU
SH
MAKE
SEG
LED
GPIO
CPU
ASM
C
C++
BOOT
MEM
LD
GCC
BL
GDB
VALGRIND
PROF


LEGO
HDMI
SENSOR
I2C
SD
PS2
SPI
GIT
MIDI
Logic Analyzer
SER
UART
GPU
SH
MAKE
SEG
LED
GPIO
CPU
ASM
C
C++
Java
BOOT
MEM
LD
GCC
BL
GDB
VALGRIND
PROF

Goal 2

Know Your Tools


LEGO
HDMI
SENSOR
I2C
SD
PS2
SPI
GIT
MIDI
Logic Analyzer
SER
UART
GPU
SH
MAKE
SEG
LED
GPIO
CPU
ASM
C
C++
Java
BOOT
MEM
LD
GCC
BL
GDB
VALGRIND
PROF


To invent, you need a good imagination and a pile of junk
Thomas Edison

Another Goal

Debugging, Testing, and Troubleshooting

Engineering Habits

Start from a working, known state

Make things visible (printf and logic analyzer)

Methodical (D&C), not random, search for problem

Form hypotheses and perform experiments

Fast prototyping loop, automate, … (1-click build)

Heed warnings produced by tools

Don’t become frustrated, relish the challenge

Clean and Simple
is
Beautiful

```
/* match: search for regexp anywhere in text */
int match(char *regexp, char *text) {
  if (regexp[0] == '^')
    return matchhere(regexp+1, text);
  do {    /* must look even if string is empty */
    if (matchhere(regexp, text))
      return 1;
  } while (*text++ != '\0');
  return 0;
}
/* matchhere: search for regexp at beginning of text */
int matchhere(char *regexp, char *text) {
  if (regexp[0] == '\0')
    return 1;
  if (regexp[1] == '*')
    return matchstar(regexp[0], regexp+2, text);
  if (regexp[0] == '$' && regexp[1] == '\0')
    return *text == '\0';
  if (*text!='\0' && (regexp[0]=='.' || regexp[0]==*text))
    return matchhere(regexp+1, text+1);
  return 0;
}
/* matchstar: search for c*regexp at beginning of text*/
int matchstar(int c, char *regexp, char *text) {
  do { /* a * matches zero or more instances */
    if (matchhere(regexp, text))
      return 1;
  } while (*text != '\0' && (*text++ == c || c == '.'));
  return 0;
}
```

Ken Thompson

Regular Expression

ch

.

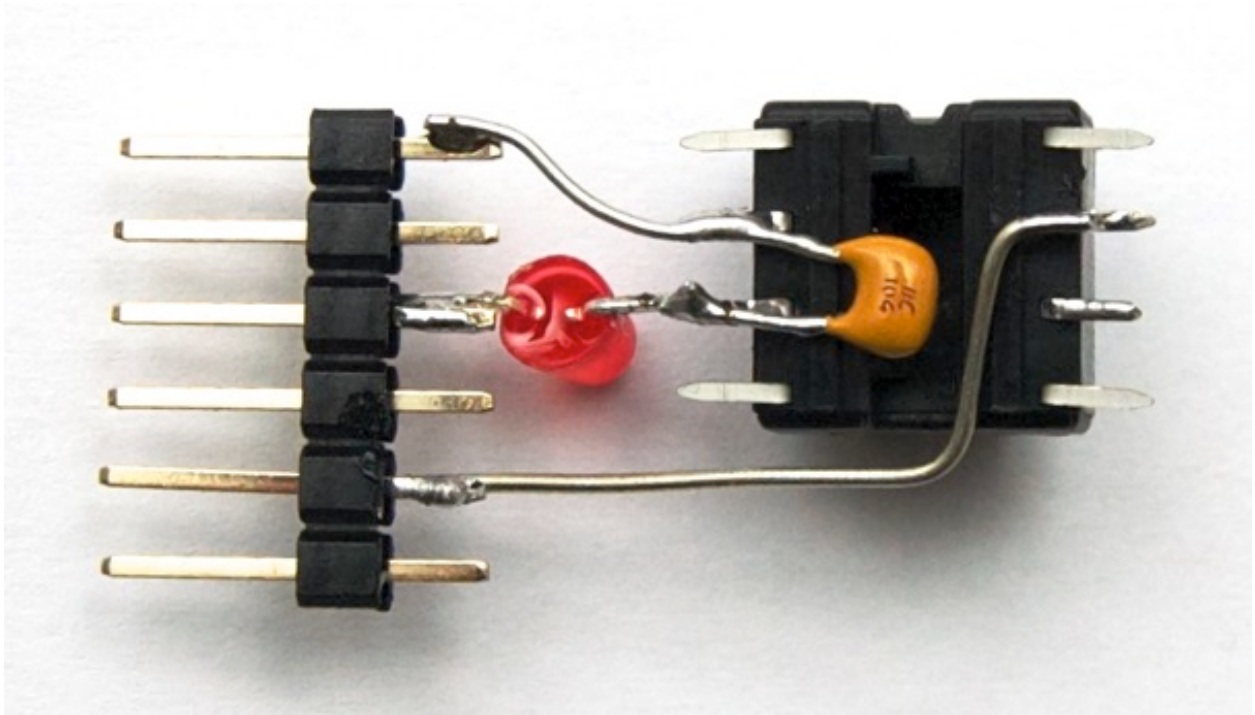
*

^

$

Read *Beautiful Code*

Jean-Claude Wippler



www.jeelabs.org

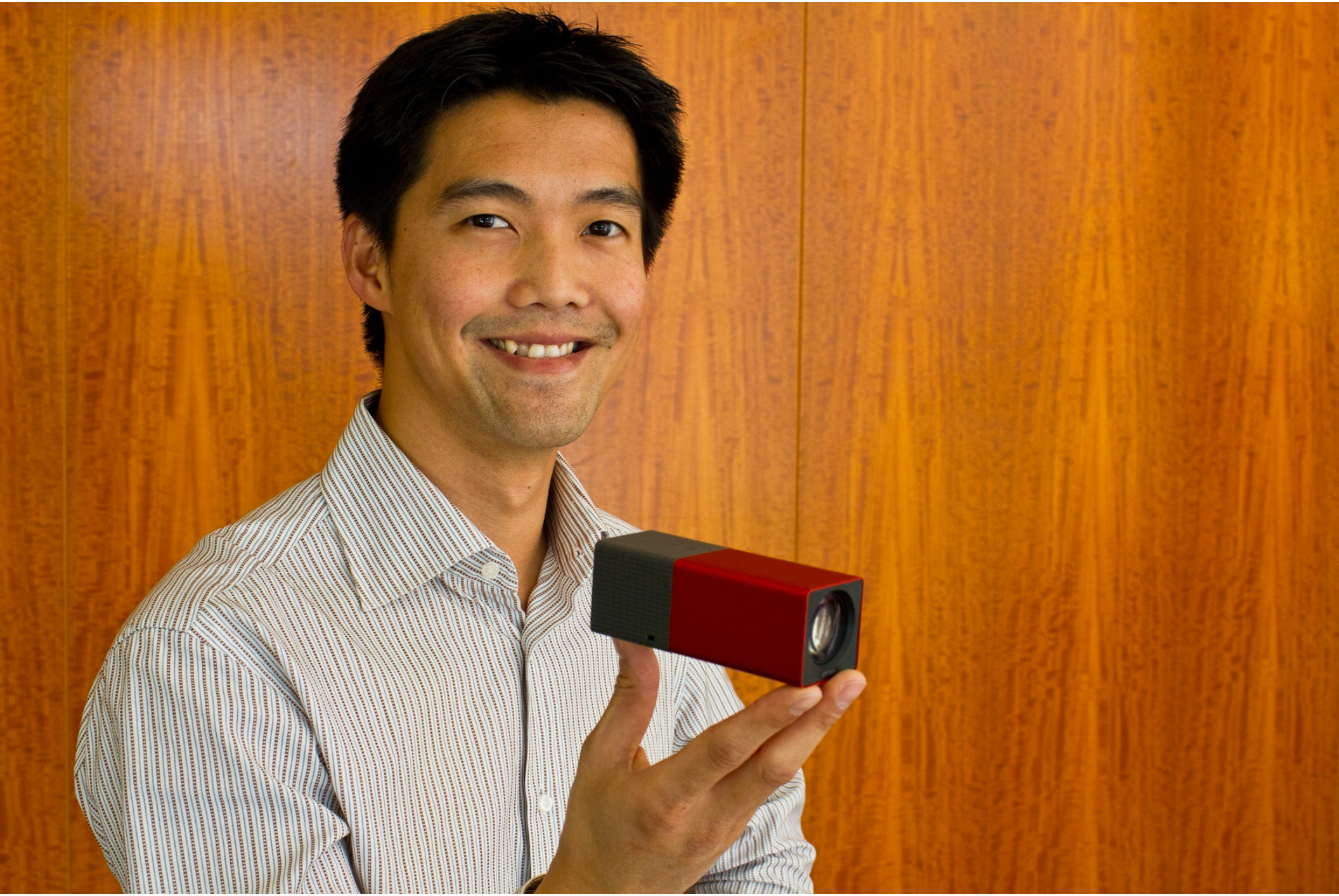
Be a Maker and a Doer

Build Systems

Light Field Camera

(A) (B) (C)

Figure 3.10: The medium format digital camera used in our prototype.



Linux and Beyond

Modes

Privileged modes

Exception modes

User	System	Supervisor	Abort	Undefined	Interrupt	Fast interrupt
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	R8_fiq
R9	R9	R9	R9	R9	R9	R9_fiq
R10	R10	R10	R10	R10	R10	R10_fiq
R11	R11	R11	R11	R11	R11	R11_fiq
R12	R12	R12	R12	R12	R12	R12_fiq
R13	R13	R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
R14	R14	R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
PC	PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

indicates that the normal register used by User or System mode has been replaced by an alternative register specific to the exception mode

Modes

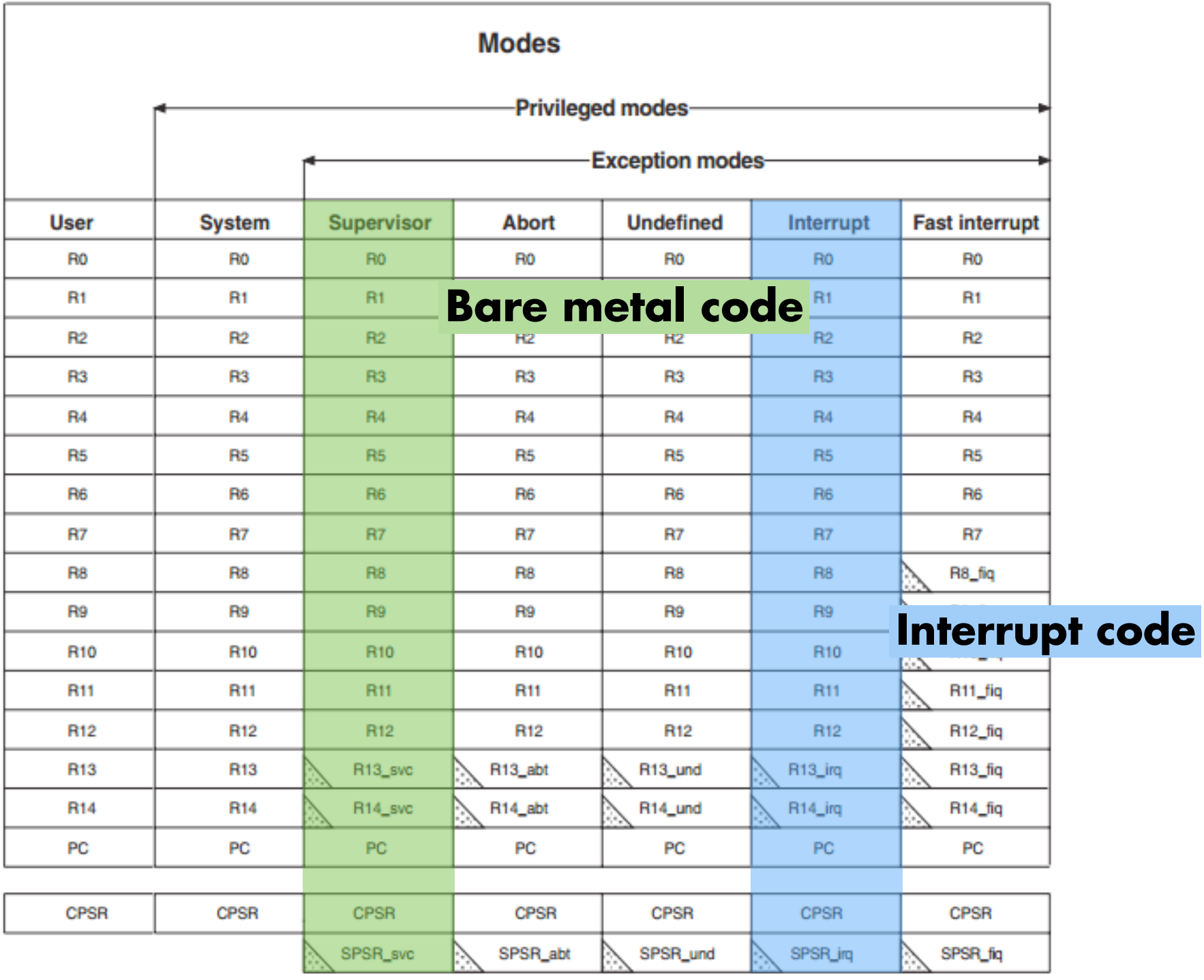
Privileged modes

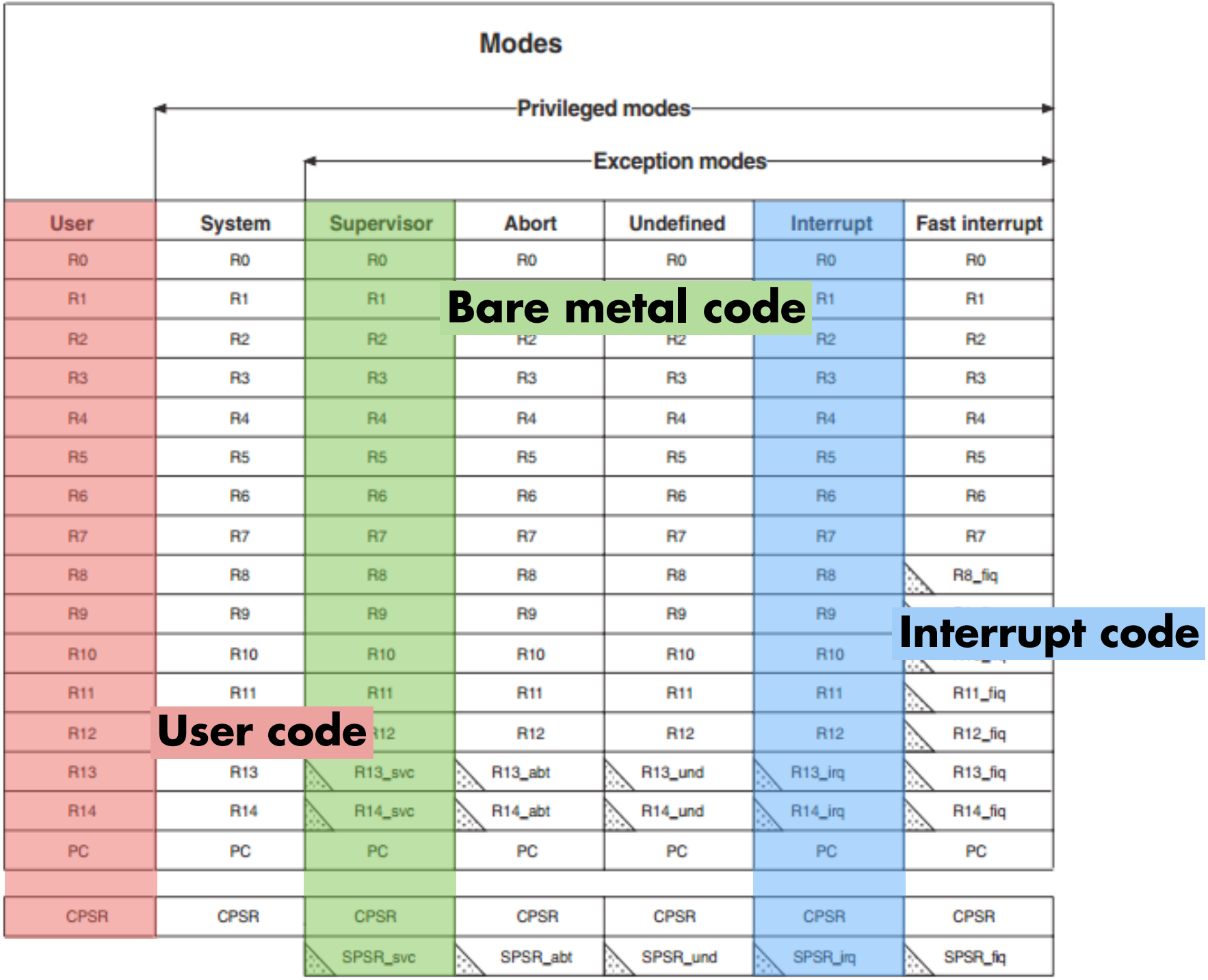
Exception modes

User	System	Supervisor	Abort	Undefined	Interrupt	Fast interrupt
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1			R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	R8_fiq
R9	R9	R9	R9	R9	R9	R9_fiq
R10	R10	R10	R10	R10	R10	R10_fiq
R11	R11	R11	R11	R11	R11	R11_fiq
R12	R12	R12	R12	R12	R12	R12_fiq
R13	R13	R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
R14	R14	R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
PC	PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

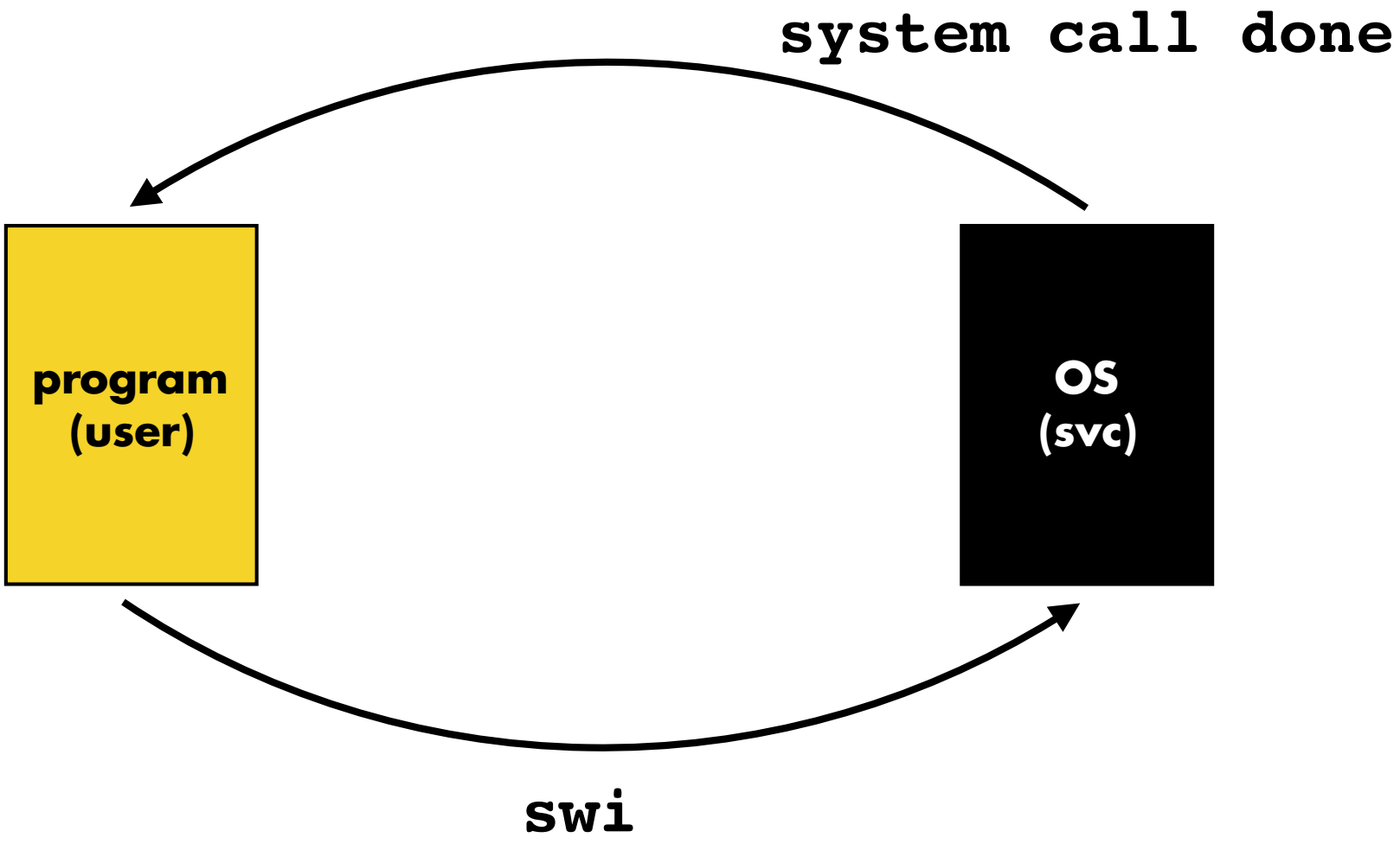
Bare metal code

indicates that the normal register used by User or System mode has been replaced by an alternative register specific to the exception mode


Modes
Privileged modes
Exception modes
User
System
Supervisor
Abort
Undefined
Interrupt
Fast interrupt
R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
PC
CPSR
SPSR_svc
SPSR_abt
SPSR_und
SPSR_irq
SPSR_fiq
R8_fiq
R9_fiq
R10_fiq
R11_fiq
R12_fiq
R13_svc
R13_abt
R13_und
R13_irq
R13_fiq
R14_svc
R14_abt
R14_und
R14_irq
R14_fiq
Bare metal code
Interrupt code
indicates that the normal register used by User or System mode has been replaced by an alternative register specific to the exception mode


Modes
Privileged modes
Exception modes
User
System
Supervisor
Abort
Undefined
Interrupt
Fast interrupt
R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
PC
CPSR
SPSR
R8_fiq
R9_fiq
R10_fiq
R11_fiq
R12_fiq
R13_svc
R13_abt
R13_und
R13_irq
R13_fiq
R14_svc
R14_abt
R14_und
R14_irq
R14_fiq
SPSR_svc
SPSR_abt
SPSR_und
SPSR_irq
SPSR_fiq
Bare metal code
User code
Interrupt code
indicates that the normal register used by User or System mode has been replaced by an alternative register specific to the exception mode

Operating System



User mode

User mode can't do a lot of things...

1. **Modify CPSR**
2. **Access configuration registers**
3. **Access IO registers/memory**

Hardware throws an exception if it tries

Enables operating system to keep processes from interfering with each other

Operating Systems

Implementing device drivers

Scheduling processes and tasks

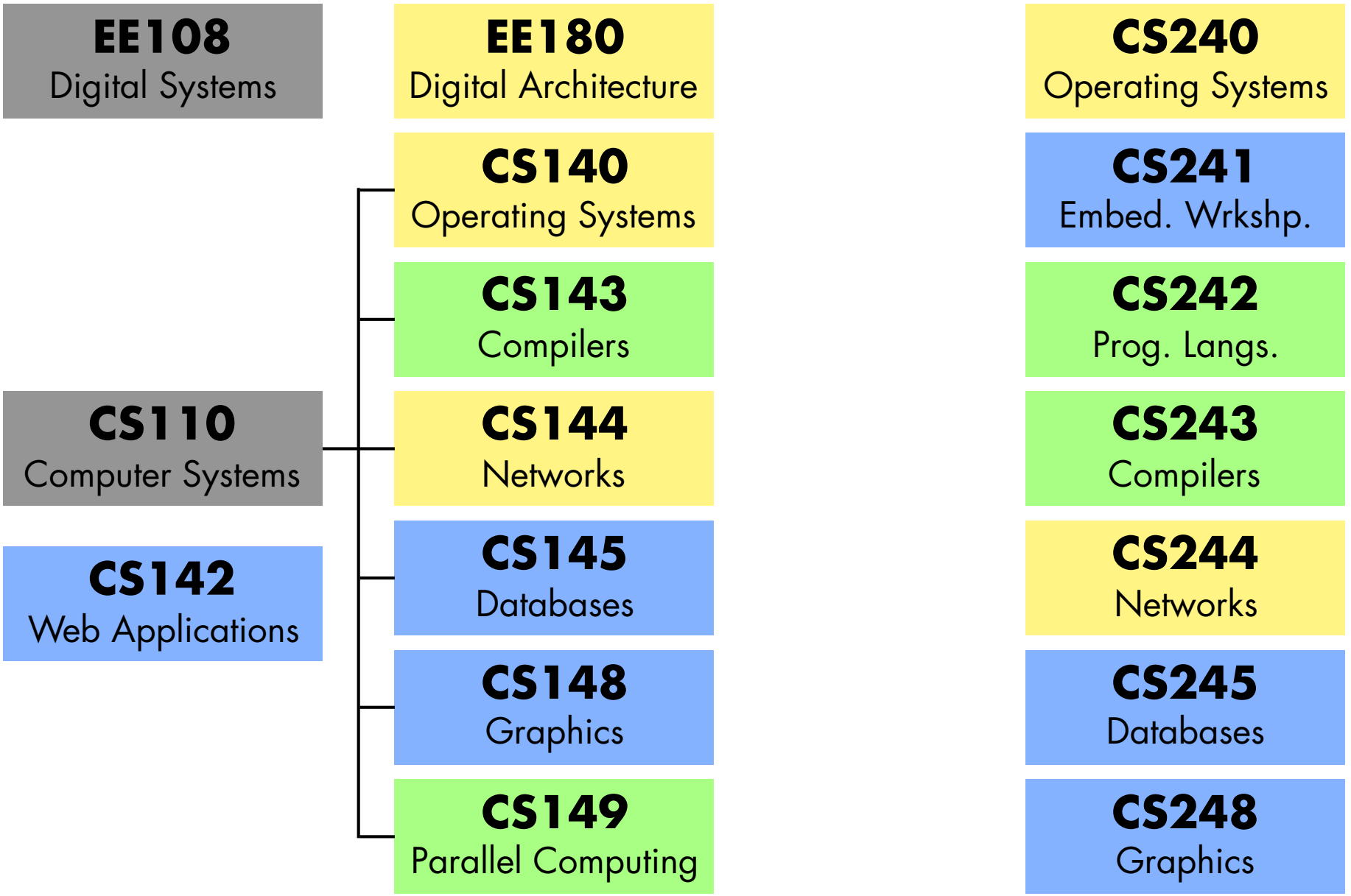
Protecting processes from each other

Virtual memory

File systems storage

Networking

Next Steps


EE108
Digital Systems
CS110
Computer Systems
CS142
Web Applications
EE180
Digital Architecture
CS140
Operating Systems
CS143
Compilers
CS144
Networks
CS145
Databases
CS148
Graphics
CS149
Parallel Computing
CS240
Operating Systems
CS241
Embed. Wrkshp.
CS242
Prog. Langs.
CS243
Compilers
CS244
Networks
CS245
Databases
CS248
Graphics

Demonstrations

- **Mon at 12:15 in Gates 325**
- **Start with fast forward : 2 minute project pitch**
- **Setup project and demo to students and instructors**

Submission

- **Due Wed at midnight**
- **Final github commit**
- **Include README.md describing your project (pictures!!, attributions)**

Logistics

- **Return keyboards (the rest of the stuff is yours)**
- **Submit reimbursements**

Comments and Suggestions?