情報連携学部

INIAD

INIAD Computer Architecture Practice

第1回 CPUの命令セット
CPU instruction set

1. この実習の概要
Outline of this course

このコースの目標 Goal of this course

- CPUのマシンコードを書いて実行する
  Write and execute CPU machine code.
  - CPUの命令セットを確認する
    Check the CPU instruction set.
  - 命令のシミュレータを作る
    Make an instruction simulator.
  - 命令コードの簡易アセンブラを作る
    Make a simple assembler of instruction codes.
  - マシンコードを書いてそのコードの実行をシミュレーションする
    Write machine code and simulate the execution of the code.

開発対象の命令セット Instruction set to be developed

- M32R命令セット M32R instruction set
  - CA講座のCPU16命令セットの出典
    It is the source of the CPU16 instruction set in CA course.
    - CPU16は16ビットCPUだったが、M32Rは32ビットCPU
      CPU16 was a 16-bit CPU, but M32R is a 32-bit CPU.
  - 参考資料 Reference documents
    - 日本語:
      https://www.renesas.com/jp/ja/doc/products/mpumcu/002/rjj09b0107_32fpusm.pdf
      •この資料の3-112ページ、「②STH Rsrc1,@Rsrc2+ [M32R-FPU拡張ニーモニック]」は、このコースでの開発の対象外です。無視してください。
      •Page 3-112 of this document, “②STH Rsrc1, @ Rsrc2 + [M32R-FPU Extended Mnemonic]”, is not covered by this course. Please ignore.
    - English:
      https://1jzmerc.com/files/pdf/mcu/m3217x-e32fpu.pdf

CPU16命令セット CPU16 instruction set

Type	Name	Num. of bits	1st 16-bit code																2nd 16-bit code (optional)														Funcition
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Load / Store	LD	16	0	0	1	0	d				1	1	0	0	s																				Load register indirect
	LD	32	1	0	1	0	d				1	1	0	0	s				disp16																Load register relative indirect
	ST	16	0	0	1	0	s1				0	1	0	0	s2																				Store register indirect
	ST	32	1	0	1	0	s1				0	1	0	0	s2				disp16																Store register relative indirect
Data transfer	MV	16	0	0	0	1	d				1	0	0	0	s																				Move
	LDI	32	1	0	0	1	d				1	1	1	1	0	0	0	0	imm16																Move immediate constant
Arithmetic	ADD	16	0	0	0	0	d				1	0	1	0	s																				Add
	ADDI	16	0	1	0	0	d				imm8																								Add immediate constant
	SUB	16	0	0	0	0	d				0	0	1	0	s																				Subtract
	AND	16	0	0	0	0	d				1	1	0	0	s																				Bitwise AND
	OR	16	0	0	0	0	d				1	1	1	0	s																				Bitwise OR
	XOR	16	0	0	0	0	d				1	1	0	1	s																				Bitwise XOR
	NOT	16	0	0	0	0	d				1	0	1	1	s																				Bitwise NOT
	MUL	16	0	0	0	1	d				0	1	1	0	s																				Multiply
	CMP	16	0	0	0	0	s1				0	1	0	0	s2																				Compare and set C-bit
Jump	BRA	16	0	1	1	1	1	1	1	1	pcdisp8																								Branch
	BC	16	0	1	1	1	1	1	0	0	pcdisp8																								Branch on C-bit
	BNC	16	0	1	1	1	1	1	0	1	pcdisp8																								Branch on not C-bit
	JMP	16	0	0	0	1	1	1	1	1	1	1	0	0	s																				Jump
	JL	16	0	0	0	1	1	1	1	0	1	1	0	0	s																				Jump and link

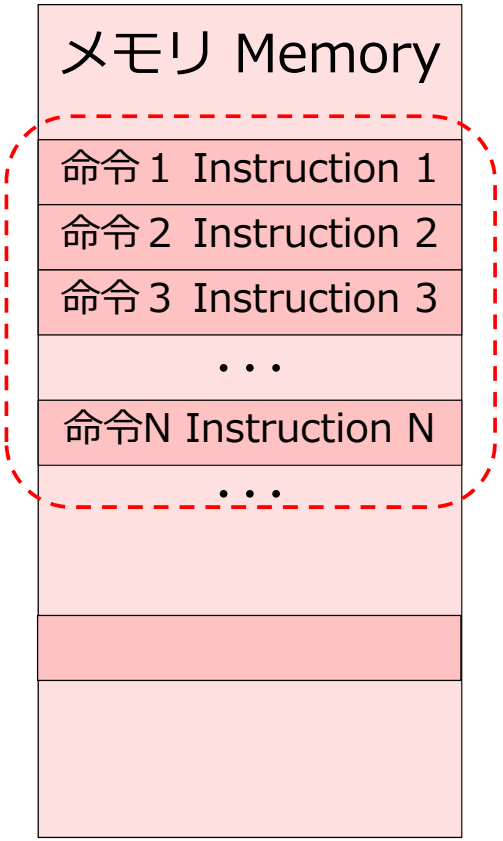
2. マシンコードと命令
Machine code and instruction

Cで書かれたプログラムの実行
Execution of a program written in C

- C言語で書かれたプログラム A program in C language
  ↓ C compiler (a software)が変換 Translates it.
- そのプログラムのマシンコード Machine code of the program
  Processor (a hardware)が実行 Executes it.
- プロセッサはマシンコードしか実行できない
  Processor can only execute machine code.

マシンコード Machine code

- マシンコードとは？
  What is the machine code?
  - メモリに格納された2進コード
    A binary code stored in memory.
  - 命令コード（命令）の列
    A series of instruction codes (instructions).
  - 1つの命令はCPUの動作の1単位を示す
    An instruction specifies a unit of the CPU operation.



マシンコードの生成 Machine code generation

- Cで書かれたソースコードからマシンコードを生成する
  Generate machine code from a source code written in C
- 例 Example: factorial.c
  - nの階乗 (n!) を計算する Calculate the factorial of n (n!)
    n! = n×(n-1)×(n-2)×…×2×1

```
int factorial(int n)
{
  if (n == 1)
    return 1;
  return n*factorial(n - 1);
}
```

マシンコードの生成 Machine code generation

- C compilerでソースコードをマシンコードに変換する
  Translate source code into machine code with C compiler
  - ソースコードをコンパイルする Compile the source code
    >gcc -c factorial.c –o factorial.o –g
  - 生成されたマシンコード（2進データ）を人が読める形式で表示する
    Display the generated machine code (binary data) in human readable form
    >objdump -S factorial.o

マシンコードの生成 Machine code generation

- 1行が1命令に対応
  One line corresponds to one instruction.
  - アドレス Address
  - 2進コード Binary code
  - ニモニック記述 Mnemonic description
    - ニモニック記述の為のプログラミング言語を「アセンブリ言語」と呼ぶ
      The programming language for mnemonic description is called “assembly language”.
    - アセンブリ言語で書かれた記述をマシンコードに変換するツールを「アセンブラ」と呼ぶ
      A tool that converts a description written in assembly language into machine code is called an “assembler”.

```
00000000 <_factorial>:
int factorial(int n)
{
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   83 ec 18                sub    $0x18,%esp
 if (n == 1)
   6:   83 7d 08 01             cmpl   $0x1,0x8(%ebp)
   a:   75 07                   jne    13 <_factorial+0x13>
    return 1;
   c:   b8 01 00 00 00          mov    $0x1,%eax
  11:   eb 12                   jmp    25 <_factorial+0x25>
 return n*factorial(n - 1);
  13:   8b 45 08                mov    0x8(%ebp),%eax
  16:   83 e8 01                sub    $0x1,%eax
  19:   89 04 24                mov    %eax,(%esp)
  1c:   e8 df ff ff ff          call   0 <_factorial>
  21:   0f af 45 08             imul   0x8(%ebp),%eax
}
  25:   c9                      leave
  26:   c3                      ret
  27:   90                      nop
```

3. 本日の課題
Today’s challenge

命令表を完成する Complete the instruction table

- 命令表 Instruction table
  - 命令名 Instruction name
    - このプロジェクトでサポートする命令を挙げた。
      Listed the instructions supported by this project.
    - 命令コードフォーマットの違いに応じて命令名を変えた。
      Changed the instruction name according to the difference of the instruction code format.
  - バイト数 Bytes
    - 命令コードのバイト数
      The number of bytes of the instruction code.
  - Opコード Opcode
    - オペランドフィールドには0を埋めてある。
      The operand field is filled with 0.

	A	B	C	D
1	**Instruction name**	**Bytes**	**Opcode**	
2	**add**	**2**	**00a0**	
3	add3			
4	**addi**	**2**	**4000**	
5	**and**	**2**	**00c0**	
6	and3			
7	**bc**	**2**	**7c00**	
8	bc24 (BC 24-bit disp)			
9	beq			
10	beqz			
11	bgez			
12	bgtz			
13	bl			
14	bl24 (BL 24-bit disp)			
15	blez			
16	bltz			
17	**bnc**	**2**	**7d00**	
18	bnc24 (BNC 24-bit disp)			
19	bne			
20	bnez			
21	**bra**	**2**	**7f00**	
22	bra24 (BRA 24-bit disp)			
23	**cmp**	**2**	**0040**	
24	cmpi			
25	cmpu			

命令表を完成する Complete the instruction table

- 命令表を参照して、シミュレータは右のように書かれている
  Referring to the instruction table, the simulator is written as shown on the right.

```
switch( icode1&0xf0f0 ) {
case 0x00a0: // ADD
case 0x00c0: // AND
case 0x0040: // CMP
case 0x20c0: // LD
case 0xa0c0: // LDR
case 0x1060: // MUL
case 0x1080: // MV
case 0x00b0: // NOT
case 0x00e0: // OR
case 0x2040: // ST
case 0xa040: // STR
case 0x0020: // SUB
case 0x00d0: // XOR
default:
    switch( icode1&0xff00 ) {
    case 0x7c00: // BC
    case 0xfc00: // BC24
    case 0x7d00: // BNC
    case 0xfd00: // BNC24
    case 0x7f00: // BRA
    case 0xff00: // BRA24
    default:
        switch( icode1&0xfff0 ) {
        case 0x1ec0: // JL
        case 0x1fc0: // JMP
        default:
            if( (icode1&0xf000) == 0x4000 ) { // ADDI
            } else if( (icode1&0xf000) == 0x6000 ) { // LDI
            } else if( (icode1&0xf0ff) == 0x90f0 ) { // LDI16
            } else if( (icode1&0xffff) == 0x7000 ) { // NOP
            } else {
                printf("Unknown instruction: icode1 = %04x¥n", icode1);
                return -1;
```

<詳細は省略する。 The details are omitted.>

命令表を完成する Complete the instruction table

- 命令表のテンプレートは次のフォルダに置かれている
  The instruction table template is located in the following folder:
  - https://drive.google.com/drive/u/1/folders/1G6_Ma4Wy1ld4bjC64KEaXkaKgWDUUJ6q
  - 命令表 Instruction table
    isa32r.xlsx
  - シミュレータコード Simulator code
    sim32r.c