
Intel 32bit アーキテクチャー

コンピュータ基礎 (8)

菊池浩明

1. Intelアーキテクチャー

- 8080
 - 8 bit, 1974
- _____
 - 16bit, 1978, x86
- 80386
 - 32bit, 1985, IA-32
- Core-2
 - 64bit, 2006, x64

プロセッサ構成要素

- ALU
- Register
- Bus

レジスター register

■ 8086 (16bit)

- ❑ AX Accumulator reg.
- ❑ BX Base address
- ❑ CX Count
- ❑ DX Data
- ❑ SI Source
- ❑ DI Destination

■ 80386 (IA32)

- ❑ EAX (Extended)
- ❑ EBX
- ❑ ECX
- ❑ EDX
- ❑ ESI
- ❑ EDI
- ❑ ESP (Stack)
- ❑ EBP (Stack Flame)

32-bit 16-bitレジスタの関係

AX (16 bit)



EAX (64)

BX (16 bit)



EBX (64)

フラグ

- 主なFlag register
 - OF Overflow
 - DF Direction
 - SF Sign Flag
 - ZF Zero Flag
 - PF Parity Flag
 - CF Carry Flag

セグメント

- 有効アドレス Effective Address

- 有効addr (20-bit)
= Segment address (16 + 4bit)
+ Offset addr (8086, 16bit
の例)

- 例) CS = 16AD, Offset = 0100の時,
IP = 16AD0 + 0100 = _____

- セグメントレジスター

- DS Data Segment
 - CS Code Segment
 - ES Extra Segment
 - SS Stack Segment

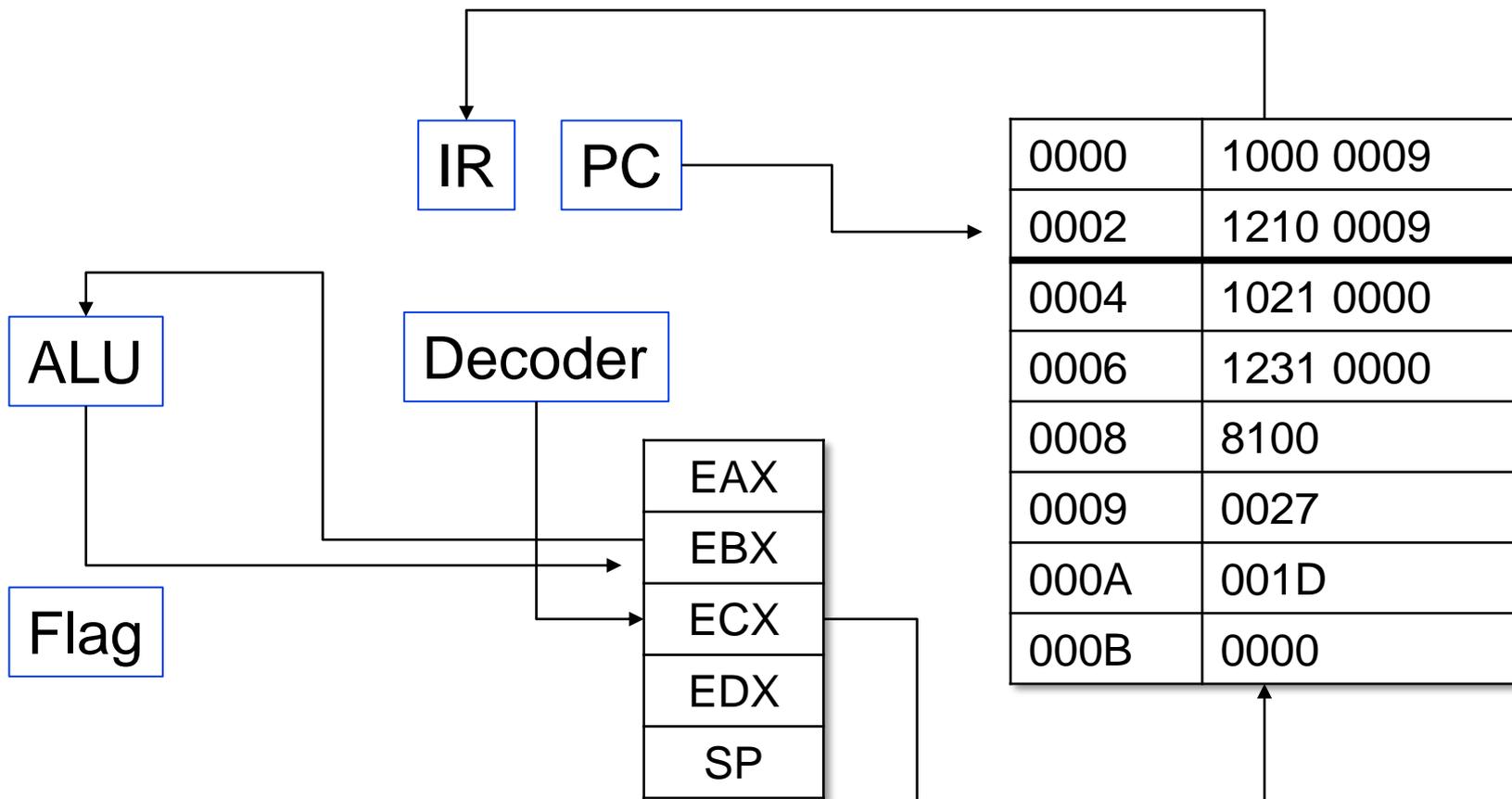
アドレス空間

CPU	アドレスバス	例	メモリ容量
8 bit	16 bit	8085, Z-80	64 KB (2^{16})
16 bit		8086	1MB (2^{20})
32 bit	32 bit	Pentium D	
64 bit	64 bit	Core 2 Duo, I 7	16EB (2^{64})

2. 動作の流れ

- 実行ステップ
 - 1. 命令読み出し ()
 - 2. 命令解読 (decode)
 - 3. 実行 (execute)
 - 4. 結果格納 (write-back)

命令の処理

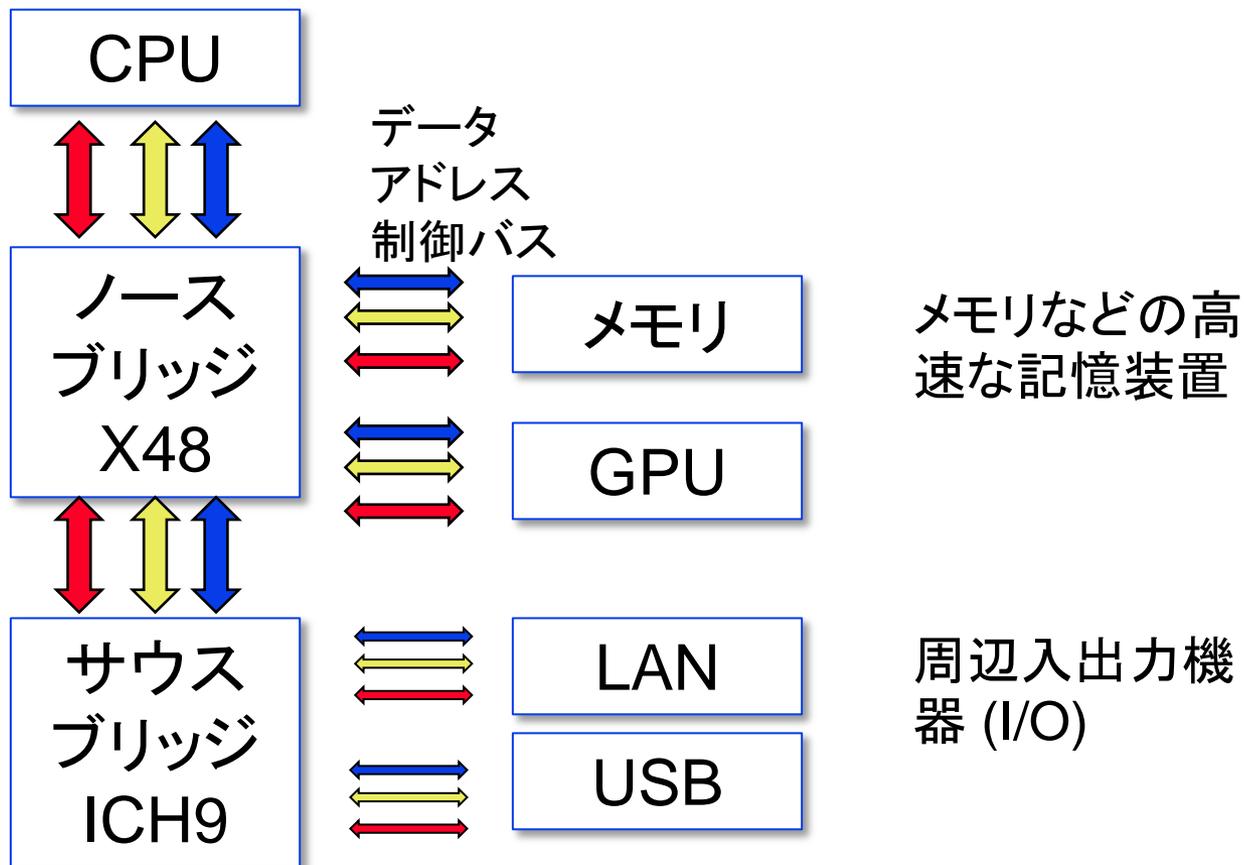


Bus

- Bus (8086の例) 双方向データ信号線
 - アドレスバス A_0-A_{19}
 - データバス A_0-A_{15}
 - 制御バス S_0-S_7

時刻	1	2	3	4	5	6	7	8	9
$S_2 S_1 S_0$	100	101	000	110	100	101	000	110	100
バス	アド レス	命令		デー タ	アド レス	命令		デー タ	アド レス
内容	Fetc h	read	(exe)	write	Fetc h	read	(exe)	Fetc h	Fetc h

コントロールバス



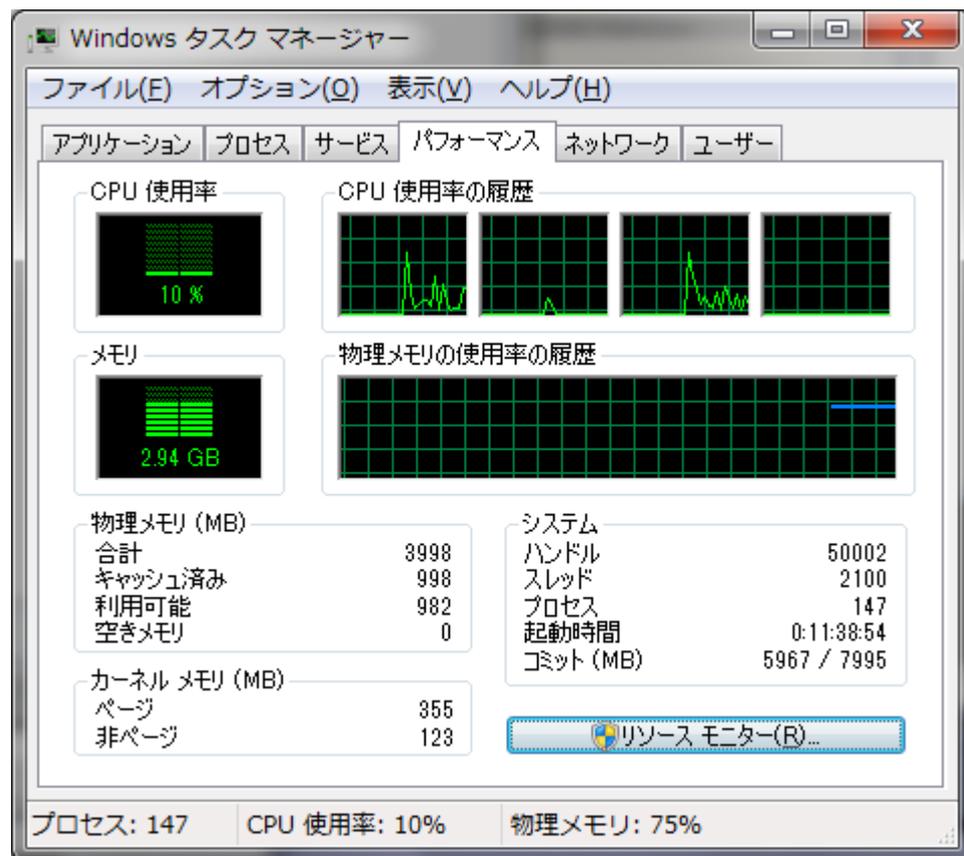
高速化の工夫

- キャッシュメモリ
- パイプライン
- スーパースカラー
- マルチコア
- VLIW

マルチコア

■ Core

- Fetch, Decode, Execute, Write-backなどの回路（キャッシュはない）
- 単一のCPUに複数のコア = マルチコア
- Core 2以降



3. メモリーダンプ

■ Stirling (バイナリエディター)

```
<table border="0"
cellpadding="0"
width="800">
  <tr>
    <td><div id="table-
left"><h2><a
href="#"></a></h2></div></td>
```

ADDRESS	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
00000270	6F	72	61	74	6F	72	79	20	3C	2F	68	31	3E	0D	0A	0D	oratory </h1>...
00000280	0A	3C	74	61	62	6C	65	20	62	6F	72	64	65	72	3D	22	.<table border="
00000290	30	22	20	63	65	6C	6C	70	61	64	64	69	6E	67	3D	22	0" cellpadding="
000002A0	30	22	20	63	65	6C	6C	73	70	61	63	69	6E	67	3D	22	0" cellspacing="
000002B0	30	22	20	77	69	64	74	68	3D	22	38	30	30	22	3E	0D	0" width="800">.
000002C0	0A	20	20	3C	74	72	3E	0D	0A	20	20	20	20	3C	74	64	. <tr>.. <td
000002D0	3E	3C	64	69	76	20	69	64	3D	22	74	61	62	6C	65	2D	><div id="table-
000002E0	6C	65	66	74	22	3E	3C	68	32	3E	3C	61	20	68	72	65	left"><h2><a href
000002F0	66	3D	22	23	22	3E	3C	69	6D	67	20	73	72	63	3D	22	f="#"></h2
00000340	3E	3C	2F	64	69	76	3E	3C	2F	74	64	3E	0D	0A	20	20	></div></td>..
00000350	20	20	3C	74	64	3E	3C	64	69	76	20	69	64	3D	22	74	<td><div id="t
00000360	61	62	6C	65	2D	72	69	67	68	74	22	3E	3C	62	3E	20	able-right">
00000370	4D	65	69	6A	69	20	55	6E	69	76	65	72	73	69	74	79	Meiii University

EXEファイル

D:\¥Kikuchi¥Wrk¥maem>addval
1+2 = 3

- addval.exe (2,560 byte)

```
addval.exe
ADDRESS 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 0123456789ABCDEF
00000000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ.....
00000010 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ク.....@.....
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000030 00 00 00 00 00 00 00 00 00 00 00 00 C0 00 00 00 .....タ...
00000040 0E 1E BA 0E 00 BA 09 CD 21 B8 01 4C CD 21 54 68   エ  タ  ク  I  A  L  T  b
00000050
addval.exe
ADDRESS 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 0123456789ABCDEF
00000060 00000750 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000070 00000760 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000080 00000770 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000090 00000780 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000A0 00000790 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000B0 000007A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000C0 000007B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000D0 000007C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000E0 000007D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000F0 000007E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000100 000007F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000110 00000800 31 2B 32 20 3D 20 00 00 00 00 00 00 00 00 00 00 1+2 = .....
00000120 00000810 00 00 00 00 00 00 00 00 00 00 00 00 0D 0A 00 00 .....
00000130 00000820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000140 00000830 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000150 00000840 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

逆アセンブラー OllyDbg

The screenshot displays the OllyDbg interface for the process 'addval.exe'. The main window shows assembly code for the 'CPU - main thread, module ntdll'. The code is as follows:

```
776A0000 8B4424 04 MOV EAX, DWORD PTR SS:[ESP+4]
776A0004 CC INT3
776A0005 C2 0400 RETN 4
776A0008 CC INT3
776A0009 90 NOP
776A000A C3 RETN
90 NOP
CC INT3
C3 RETN
90 NOP
90 NOP
776A0010 8B4C24 04 MOV ECX, DWORD PTR SS:[ESP+4]
776A0014 F641 04 06 TEST BYTE PTR DS:[ECX+4], 6
776A0018 74 05 JE SHORT ntdll.776A001F
776A001A E8 A11D0100 CALL ntdll.776A001F
776A001F B8 01000000 MOV EAX, 1
776A0024 C2 1000 RETN 10
776A0027 90 NOP
776A0028 8D8424 DC020000 LEA EAX, DWORD PTR SS:[ESP+20C]
776A002F 64:8B0D 00000000 MOV ECX, DWORD PTR FS:[0]
776A0036 BA 10006A77 MOV EDX, ntdll.776A0010
776A003E 8908 MOV DWORD PTR DS:[EAX], ECX
776A003D 8950 04 MOV DWORD PTR DS:[EAX+4], EDX
776A0040 64:A3 00000000 MOV DWORD PTR FS:[0], EAX
776A0046 58 POP EAX
776A0047 8D7C24 0C LEA EDI, DWORD PTR SS:[ESP+C]
776A004E FFD0 CALL EBX
776A004D 8B5F CC020000 MOV ECX, DWORD PTR DS:[EDI+2CC]
776A0053 64:890D 00000000 MOV DWORD PTR FS:[0], ECX
776A005A 6A 01 PUSH 1
776A005C 57 PUSH EDI
```

Registers (FPU):

EAX	00401000	addval.<ModuleEntryPoint>
ECX	00000000	
EDX	00000000	
EBX	7EFDE000	
ESP	0018FFF0	
EBP	00000000	
ESI	00000000	
EDI	00000000	
EIP	776A01B8	ntdll.776A01B8

Flags:

C	0	ES	002B	32bit	0(FFFFFFFF)
P	0	CS	0023	32bit	0(FFFFFFFF)
A	0	SS	002B	32bit	0(FFFFFFFF)
Z	0	DS	002B	32bit	0(FFFFFFFF)
S	0	FS	0053	32bit	7EFDD000(FFF)
O	0	GS	002B	32bit	0(FFFFFFFF)
T	0				
D	0				
O	0	LastErr	ERROR_SUCCESS	(00000000)	
EFL	00000202	(NO, NB, NE, A, NS, PO, GE, G)			
ST0	empty	0.0			
ST1	empty	0.0			
ST2	empty	0.0			
ST3	empty	0.0			
ST4	empty	0.0			
ST5	empty	0.0			
ST6	empty	0.0			
ST7	empty	0.0			

Memory dump (Address Hex dump ASCII):

Address	Hex	dump	ASCII
00403000	31 2B 32 20 3D 20 00 00	1+2 = ..	
00403008	00 00 00 00 00 00 00 00	
00403010	00 00 00 00 00 00 00 00	
00403018	00 00 00 00 00 0A 00 00a.....	
00403020	00 00 00 00 00 00 00 00	
00403028	00 00 00 00 00 00 00 00	
00403030	00 00 00 00 00 00 00 00	
00403038	00 00 00 00 00 00 00 00	
00403040	00 00 00 00 00 00 00 00	
00403048	00 00 00 00 00 00 00 00	
00403050	00 00 00 00 00 00 00 00	
00403058	00 00 00 00 00 00 00 00	
00403060	00 00 00 00 00 00 00 00	
00403068	00 00 00 00 00 00 00 00	
00403070	00 00 00 00 00 00 00 00	
00403078	00 00 00 00 00 00 00 00	
00403080	00 00 00 00 00 00 00 00	
00403088	00 00 00 00 00 00 00 00	
00403090	00 00 00 00 00 00 00 00	
00403098	00 00 00 00 00 00 00 00	
004030A0	00 00 00 00 00 00 00 00	
004030A8	00 00 00 00 00 00 00 00	
004030B0	00 00 00 00 00 00 00 00	
004030B8	00 00 00 00 00 00 00 00	
004030C0	00 00 00 00 00 00 00 00	

MASM

- addval.asm

```
1.          include  D:\masm32\include\masm32rt.inc
2.          .data
3.  msg db "1+2 = ", 0
4.          .code
5.  start:
6.          print OFFSET msg
7.          mov  eax, 1
8.          mov  ecx, 2
9.          add  ecx, eax
10.         print str$(ecx)
11.         print chr$(13,10)
12.         exit
13.  end start
```

CASLとIA32の比較

種類	IA32	CASL
データ転送	MOV EAX, EDX MOV EAX, 1 LEA EAX, 0123h MOV EAX, 1, EEX	LD GR0, GR1 LAD GR0, 1 LAD GR0, 0123 LD GR0, 1, GR2
データ格納	MOV 0100h, EAX	ST GR0, 0100
演算命令	ADD EAX, ECX SUB EAX, ECX CMP EAX, EBX INC EAX	ADDA GR0, GR1 SBUA GR1, GR2 CPA GR0, GR1 LAD GR0, 1, GR0
制御, 分岐命令	JZ 0100h JLE 0100h JG 0100h	JZE 0100 JMI 0100 JPL 0100

宿題

- 4章
 - 問8
 - 問9

まとめ

- インテルアーキテクチャーには, ()bitの8080から, () core i-7までの種類がある.
- EAX, EBXなどの64-bitの()と, EEX, EDXなどの()を持つ.
- 機械語の実行は, 命令の(), 読み出し, 実行, ()の主に4つのフェーズがあり, それらを並列に実行することで高速化する技術を()という.
- OllyDbgなどの機械語をアセンブラ言語に変換するツールを()という.