計算機アーキテクチャ特論 (Advanced Computer Architectures)

1. 導入：マイクロプロセッサ

吉瀬 謙二　計算工学専攻
kise _at_ cs.titech.ac.jp　www.arch.cs.titech.ac.jp
W831 講義室　木曜日　9:00 – 10:30

関連科目・履修条件等

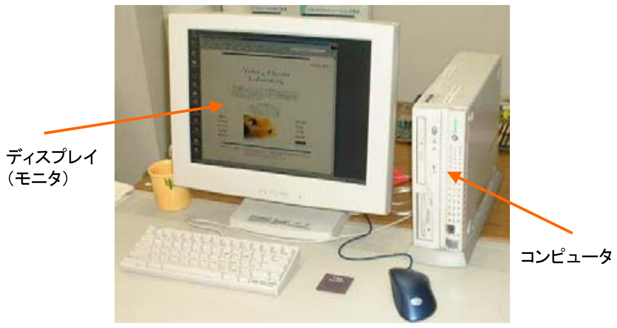
- 4学期：計算機論理設計
  - 計算機を構成するプロセッサとその制御部に関し，具体構成と設計の原理を講義する．特に，レジスタトランスファ言語を用いて計算機の内部動作を記述し，簡単な計算機の設計を行う．
- **5学期：計算機アーキテクチャ第一**
  - CPU を含め，メモリ，チャネル，入出力，通信制御，等の計算機システムを構成する各種装置について，その役割，動作原理について講義する．
- 6学期：計算機アーキテクチャ第二
  - 最新の計算機システムに採り入れられている高速プロセッサ制御方式，構成方式について述べ、これらの技術を駆使したパイプラインプロセッサ，スーパコンピュータ，超並列計算機，データフロー計算機，等の先端的なアーキテクチャについて講義する．
- 計算機アーキテクチャ特論（大学院）
  - パソコン，ワークステーション，携帯情報機器など計算機のダウンサイジング，パーソナル化に大きな役割を果たしているマイクロプロセッサについて，その動向と先端技術について講義を行う．また，演習を実施することでマイクロプロセッサ技術を習得する．

計算機アーキテクチャ特論 (Advanced Computer Architectures)

0. 導入

コンピュータ ？

コンピュータ（デスクトップ・コンピュータ）



ディスク，磁気ディスク

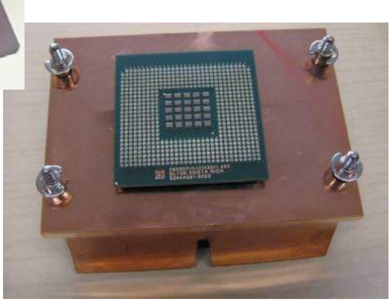
メモリ (Dynamic Random Access Memory)

7

グラフィックカード，ネットワークカード

8

マイクロプロセッサ（CPU）



9

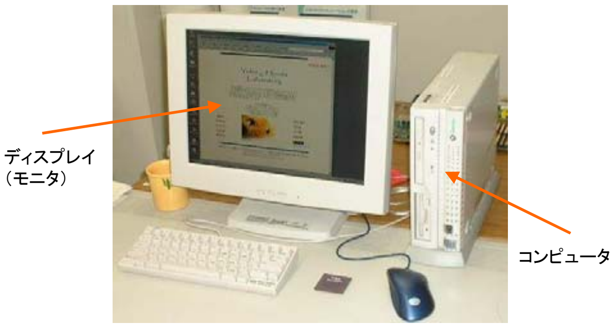
マザーボード，電源

10

コンピュータ

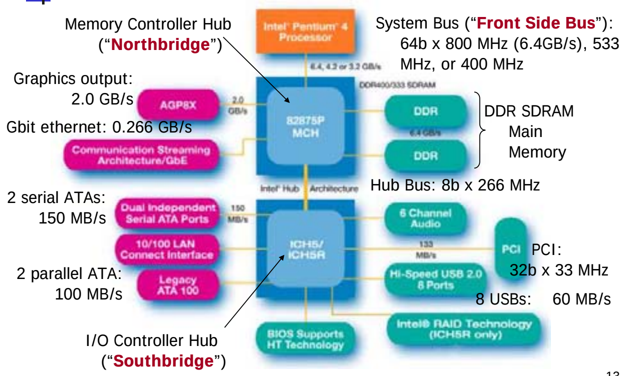
11

コンピュータ（デスクトップ・コンピュータ）



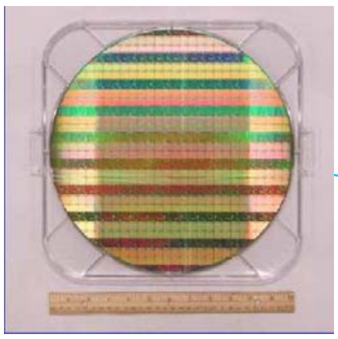
12

Example: The Pentium 4 system

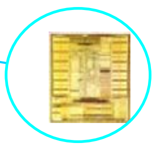

Memory Controller Hub ("Northbridge")
Intel® Pentium® 4 Processor
System Bus ("Front Side Bus"): 64b x 800 MHz (6.4GB/s), 533 MHz, or 400 MHz
6.4, 4.2 or 3.2 GB/s
Graphics output: 2.0 GB/s
AGP8X
2.0 GB/s
82875P MCH
DDR400/333 SDRAM
DDR
6.4 GB/s
DDR
DDR SDRAM Main Memory
Gbit ethernet: 0.266 GB/s
Communication Streaming Architecture/GbE
Intel® Hub Architecture
Hub Bus: 8b x 266 MHz
2 serial ATAs: 150 MB/s
Dual Independent Serial ATA Ports
150 MB/s
10/100 LAN Connect Interface
ICH5/ ICH5R
6 Channel Audio
133 MB/s
PCI
PCI: 32b x 33 MHz
2 parallel ATA: 100 MB/s
Legacy ATA 100
Hi-Speed USB 2.0 8 Ports
8 USBs: 60 MB/s
Intel® RAID Technology (ICH5R only)
BIOS Supports HT Technology
I/O Controller Hub ("Southbridge")

コンピュータアーキテクチャの魅力

プロセッサチップの製造，ウエーハとダイ



30cmのウエーハ
厚さは数ミリで、直径が30cm
大きなCDのような形をしている。



ダイ
（ウエーハから切り出した
個々のチップ）

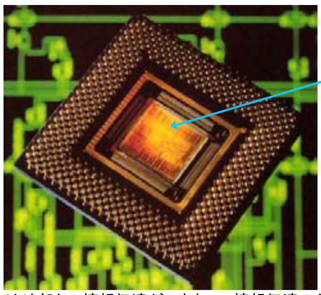
出典：Intel社, Industry-Leading Transistor Performance Demonstrated on Intel's 90-nanometer Logic Process

シリコン・インゴット，ウエーハ

Sand
Silicon Ingot
Wafer

Silicon, the most abundant element
on earth except for oxygen, is used because
it is a natural semiconductor.

プロセッサの実装，ダイのパッケージ化

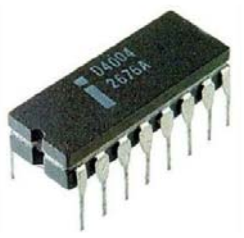


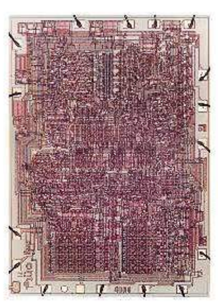
ダイのままでは外部との情報伝達ができない。情報伝達のためのピンを含む
パッケージとして加工する。

出典：Richard L. Sites, Alpha AXP Architecture Reference Manual SECOND EDITION

プロセッサを実装するためのトランジスタ

1971年: 4004 マイクロプロセッサ





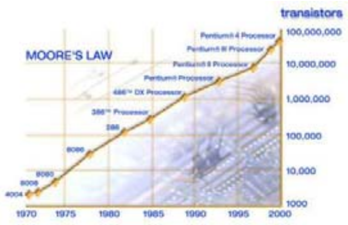
プロセッサ	出荷年	トランジスタ数
4004	1971	2,250

出典: フリー百科事典『ウィキペディア（Wikipedia）』, Intelミュージアム

ムーアの法則によるトランジスタ数の増加

ムーアの法則
チップで利用できるトランジスタの数は2年間で2倍に増加する。

プロセッサ	出荷年	トランジスタ数
4004	1971	2,250
8008	1972	2,500
8080	1974	5,000
8086	1978	29,000
286	1982	120,000
386™ processor	1985	275,000
486™ DX processor	1989	1,180,000
Pentium® processor	1993	3,100,000
Pentium II processor	1997	7,500,000
Pentium III processor	1999	24,000,000
Pentium 4 processor	2000	42,000,000

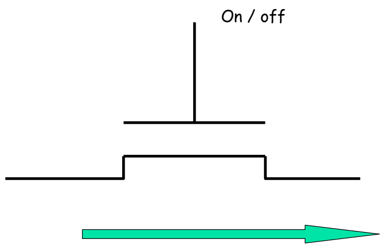


ムーアの法則に従ってトランジスタ数が増加してきた. 今後も同様の増加が見込まれる.

出典：Intel社, http://www.intel.com/research/silicon/mooreslaw.htm

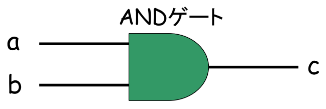
トランジスタ

- トランジスタは電気的なオン／オフ動作をするスイッチ

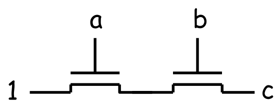


トランジスタからゲート

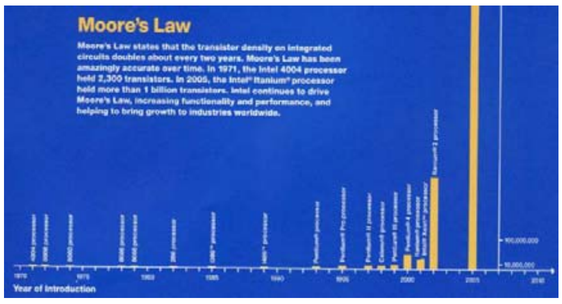
- トランジスタは電気的なオン／オフ動作をするスイッチ
- 幾つかのトランジスタから，少し機能の高いゲートを構成



a	b	c
0	0	0
1	0	0
0	1	0
1	1	1



Moore's Law

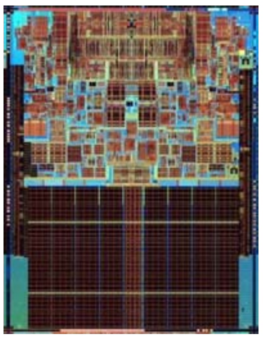


Moore's Law states that the transistor density on integrated circuits doubles about every two years. Moore's Law has been amazingly accurate over time. In 1971, the Intel 4004 processor held 2,300 transistors. In 2005, the Intel® Itanium® processor held more than 1 billion transistors. Intel continues to drive Moore's Law, increasing functionality and performance, and helping to bring growth to industries worldwide.

Moore's Law

先端マイクロプロセッサ Intel Core 2 Duo

- Core2 Duo (2006 7/27発表)
  - 65nmプロセス
  - 143mm^2
  - 291 Million トランジスタ
  - 65W
- Core Micro Architecture
  - Intelligent power capability
  - Micro-Fusion
    - RISC vs CISC
  - Advanced Smart Cache

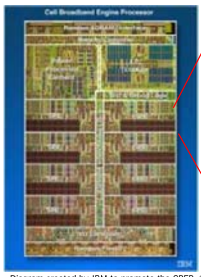


Intel Developer Forum

先端マイクロプロセッサ Cell Broadband Engine

- ヘテロジニアス チップマルチプロセッサ
  - PowerPC Processor Element (PPE) 1個
  - Synergistic Processor Element (SPE) 8個

PlayStation3 の写真は
PlaySation.com (Japan) から



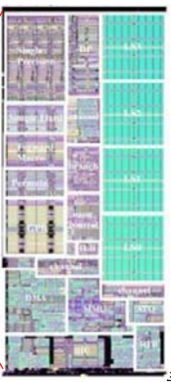
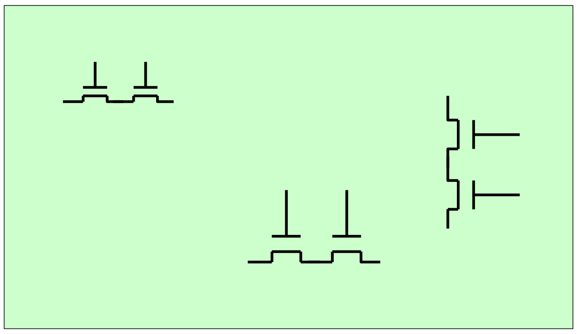


Diagram created by IBM to promote the CBEP, ©2005
WIKIPEDIAより

10億トランジスタのプロセッサ，配置，配線



10億トランジスタのプロセッサ

マルチコア（2個～数10個）からメニーコアへ

Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction, MICRO-36

数世代の
RISCプロセッサのサイズ

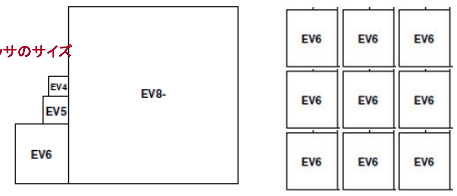
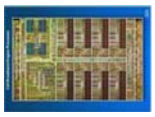


Figure 1. Relative sizes of the cores used in the study



マルチコア（2個～数10個）からメニーコアへ

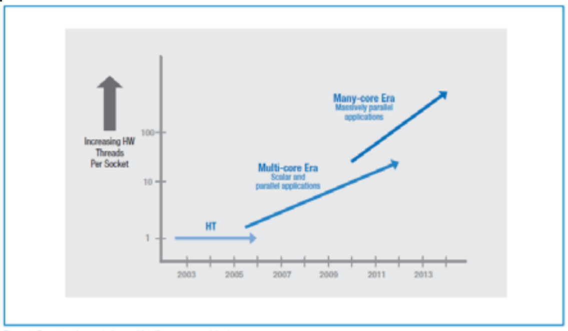


Figure 1: Current and expected eras of Intel® processor architectures

Platform 2015: Intel® Processor and Platform Evolution for the Next Decade

計算機アーキテクチャ特論
(Advanced Computer Architectures)

1. マイクロプロセッサの命令セットの例

RISC - Reduced Instruction Set Computer

- **RISC philosophy**
  - fixed instruction lengths
  - load-store instruction sets
  - limited addressing modes
  - limited operations
- Sun SPARC, HP PA-RISC, IBM PowerPC, Compaq Alpha, **MIPS**, …

Design goals: speed, cost (design, fabrication, test, packaging), size, power consumption, reliability, memory space (embedded systems)

MIPS R3000 Instruction Set Architecture (ISA)

- Instruction Categories
  - Computational
  - Load/Store
  - Jump and Branch
  - Floating Point
    - coprocessor
  - Memory Management
  - Special

Registers

R0 - R31
PC
HI
LO

3 Instruction Formats: all 32 bits wide

OP	**rs**	**rt**	**rd**	**sa**	**funct**	R format
OP	**rs**	**rt**	**immediate**			I format
OP	**jump target**					J format

MIPS Arithmetic Instructions

- MIPS assembly language **arithmetic statement**

```
add  $t0, $s1, $s2
sub  $t0, $s1, $s2
```

- Each arithmetic instruction performs only one operation
- Each arithmetic instruction fits in 32 bits and specifies exactly three operands

  destination ← source1 op source2

- Operand order is fixed (destination first)
- Those operands are all contained in the datapath's register file (`$t0,$s1,$s2`) – **indicated by $**

MIPS Register Convention, ABI (Application Binary Interface)

Name	Register Number	Usage	Preserve on call?
$zero	0	**constant 0 (hardware)**	n.a.
$at	1	reserved for assembler	n.a.
$v0 - $v1	2-3	returned values	no
$a0 - $a3	4-7	**arguments**	yes
$t0 - $t7	8-15	temporaries	no
$s0 - $s7	16-23	saved values	yes
$t8 - $t9	24-25	temporaries	no
$gp	28	global pointer	yes
$sp	29	stack pointer	yes
$fp	30	frame pointer	yes
$ra	31	return addr (hardware)	yes

Machine Language - Add Instruction

- Instructions, like registers and words of data, are **32 bits long**
- Arithmetic Instruction Format (R format):

```
add $t0, $s1, $s2
```

op	rs	rt	rd	shamt	funct

op	6-bits	opcode that specifies the operation
rs	5-bits	register file address of the first source operand
rt	5-bits	register file address of the second source operand
rd	5-bits	register file address of the result's destination
shamt	5-bits	shift amount (for shift instructions)
funct	6-bits	function code augmenting the opcode

Machine Language - Load Instruction

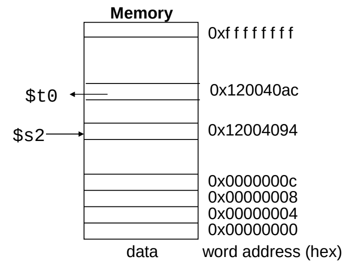
- Load/Store Instruction Format (I format):

```
lw $t0, 24($s2)
```

op	rs	rt	16 bit offset

24_{10} + $s2 =

```
  . . . 0001 1000
+ . . . 1001 0100
  . . . 1010 1100 =
        0x120040ac
```



MIPS **Memory Access** Instructions

- MIPS has two basic data transfer instructions for accessing memory

```
lw  $t0, 4($s3)  #load word from memory
sw  $t0, 8($s3)  #store word to memory
```

- The data is loaded into (lw) or stored from (sw) a register in the register file – a 5 bit address
- The memory address – a 32 bit address – is formed by adding the contents of the base address register to the offset value
  - A 16-bit field meaning access is limited to memory locations within a region of $\pm 2^{13}$ or 8,192 words ($\pm 2^{15}$ or 32,768 bytes) of the address in the base register
  - Note that the offset can be positive or negative

MIPS **Control Flow** Instructions

- MIPS conditional branch instructions:

```
bne $s0, $s1, Lbl #go to Lbl if $s0≠$s1
beq $s0, $s1, Lbl #go to Lbl if $s0=$s1
```

  - Ex:

```
        if (i==j) h = i + j;
        bne $s0, $s1, Lbl1
        add $s3, $s0, $s1
Lbl1:   ...
```

- Instruction Format (I format):

op	rs	rt	16 bit offset

- How is the branch destination address specified?

More Branch Instructions

- We have beq, bne, but what about other kinds of brances (e.g., branch-if-less-than)? For this, we need yet another instruction, slt
- Set on less than instruction:

```
slt $t0, $s0, $s1    # if $s0 < $s1    then
                     # $t0 = 1         else
                     # $t0 = 0
```

- Instruction format (R format):

op	rs	rt	rd		funct

More Branch Instructions, Con't

- Can use slt, beq, bne, and the fixed value of 0 in register $zero to create other conditions
  - less than `blt $s1, $s2, Label`

```
slt  $at, $s1, $s2      # $at set to 1 if
bne  $at, $zero, Label  #  $s1 < $s2
```

  - less than or equal to `ble $s1, $s2, Label`
  - greater than `bgt $s1, $s2, Label`
  - great than or equal to `bge $s1, $s2, Label`
- Such branches are included in the instruction set as pseudo instructions - recognized (and expanded) by the assembler
  - Its why the assembler needs a reserved register ($at)

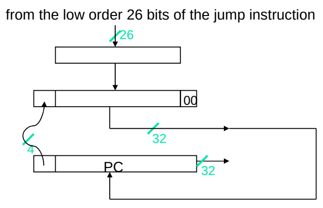
Other Control Flow Instructions

- MIPS also has an **unconditional branch** instruction or jump instruction:

```
j   label      #go to label
```

- Instruction Format (J Format):

op	26-bit address



Aside: Branching **Far Away**

- What if the branch destination is further away than can be captured in 16 bits?
  - The assembler comes to the rescue – it inserts an unconditional jump to the branch target and inverts the condition

```
          beq  $s0, $s1, L1
```

becomes

```
          bne  $s0, $s1, L2
          j    L1
     L2:
```

Instructions for Accessing Procedures

- MIPS procedure call instruction:

```
    jal  ProcedureAddress     #jump and link
```

- Saves PC+4 in register **$ra** to have a link to the next instruction for the procedure return
- Machine format (J format):

op	26 bit address

- Then can do procedure return with a

```
    jr   $ra              #return
```

- Instruction format (R format):

op	rs				funct

MIPS **Immediate** Instructions

- Small constants are used often in typical code
- Possible approaches?
  - put “typical constants” in memory and load them
  - create hard-wired registers (like $zero) for constants like 1
  - have special instructions that contain constants !

```
addi $sp, $sp, 4    #$sp = $sp + 4
slti $t0, $s2, 15   #$t0 = 1 if $s2<15
```

- Machine format (I format):

op	rs	rt	16 bit immediate

I format

- The constant is kept inside the instruction itself!
  - Immediate format limits values to the range $+2^{15}–1$ to -2^{15}

MIPS ISA So Far

Category	Instr	Op Code	Example	Meaning
Arithmetic (R & I format)	add	0 and 32	add $s1, $s2, $s3	$s1 = $s2 + $s3
	subtract	0 and 34	sub $s1, $s2, $s3	$s1 = $s2 - $s3
	add immediate	8	addi $s1, $s2, 6	$s1 = $s2 + 6
	or immediate	13	ori $s1, $s2, 6	$s1 = $s2 v 6
Data Transfer (I format)	load word	35	lw $s1, 24($s2)	$s1 = Memory($s2+24)
	store word	43	sw $s1, 24($s2)	Memory($s2+24) = $s1
	load byte	32	lb $s1, 25($s2)	$s1 = Memory($s2+25)
	store byte	40	sb $s1, 25($s2)	Memory($s2+25) = $s1
	load upper imm	15	lui $s1, 6	$s1 = 6 * 2^{16}
Cond. Branch (I & R format)	br on equal	4	beq $s1, $s2, L	if ($s1==$s2) go to L
	br on not equal	5	bne $s1, $s2, L	if ($s1 !=$s2) go to L
	set on less than	0 and 42	slt $s1, $s2, $s3	if ($s2<$s3) $s1=1 else $s1=0
	set on less than immediate	10	slti $s1, $s2, 6	if ($s2<6) $s1=1 else $s1=0
Uncond. Jump (J & R format)	jump	2	j 2500	go to 10000
	jump register	0 and 8	jr $t1	go to $t1
	jump and link	3	jal 2500	go to 10000; $ra=PC+4

MIPS Register Convention, ABI (Application Binary Interface)

Name	Register Number	Usage	Preserve on call?
$zero	0	**constant 0 (hardware)**	n.a.
$at	1	reserved for assembler	n.a.
$v0 - $v1	2-3	returned values	no
$a0 - $a3	4-7	**arguments**	yes
$t0 - $t7	8-15	temporaries	no
$s0 - $s7	16-23	saved values	yes
$t8 - $t9	24-25	temporaries	no
$gp	28	global pointer	yes
$sp	29	stack pointer	yes
$fp	30	frame pointer	yes
$ra	31	return addr (hardware)	yes

講義計画

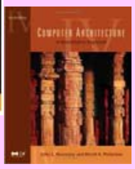
- 導入：マイクロプロセッサ
- スーパースカラプロセッサの基礎と命令レベル並列性
- 命令キャッシュ
- 分岐予測
- 動的命令スケジューリングと投機処理
- メモリデータフローとデータキャッシュ
- 組込技術，低消費電力技術
- チップマルチプロセッサ
- オンチップネットワーク，メニーコアアーキテクチャ

- 【成績評価】レポートおよび，期末レポートにより評価する．

教科書

- コンピュータアーキテクチャ 定量的アプローチ 第4版，翔泳社
- Computer Architecture, Fourth Edition: A Quantitative Approach, Fourth Edition
  - Publisher: Morgan Kaufmann; 4 edition (September 13, 2006)
  - ISBN-10: 0123704901
  - ISBN-13: 978-0123704900

ヘネシー&パターソン
コンピュータアーキテクチャ
定量的アプローチ 第4版
Computer Architecture: A Quantitative Approach 4E



参考書

- **参考書**
  - **コンピュータの構成と設計 第3版**、パターソン＆ヘネシー（成田光彰　訳）、日経BP社、2006
  - コンピュータアーキテクチャ, 村岡 洋一 著, 近代科学社, 1989
  - 計算機システム工学, 富田 真治, 村上 和彰 著, 昭晃堂, 1988
  - コンピュータハードウエア, 富田 真治, 中島 浩 著, 昭晃堂, 1995
  - 計算機アーキテクチャ, 橋本 昭洋 著, 昭晃堂, 1995
- 教科書
  - 命令レベル並列処理, 安藤秀樹 コロナ社

アナウンス

- 講義スライド，講義スケジュール
  - www.arch.cs.titech.ac.jp