計算機システム

教授　石川　裕
助教　美添
連絡先：ishikawa@is.s.u-tokyo.ac.jp
yoshizoe@is.s.u-tokyo.ac.jp
本郷　理学部7号館505、507

計算機システム　第3回

2010/11/1

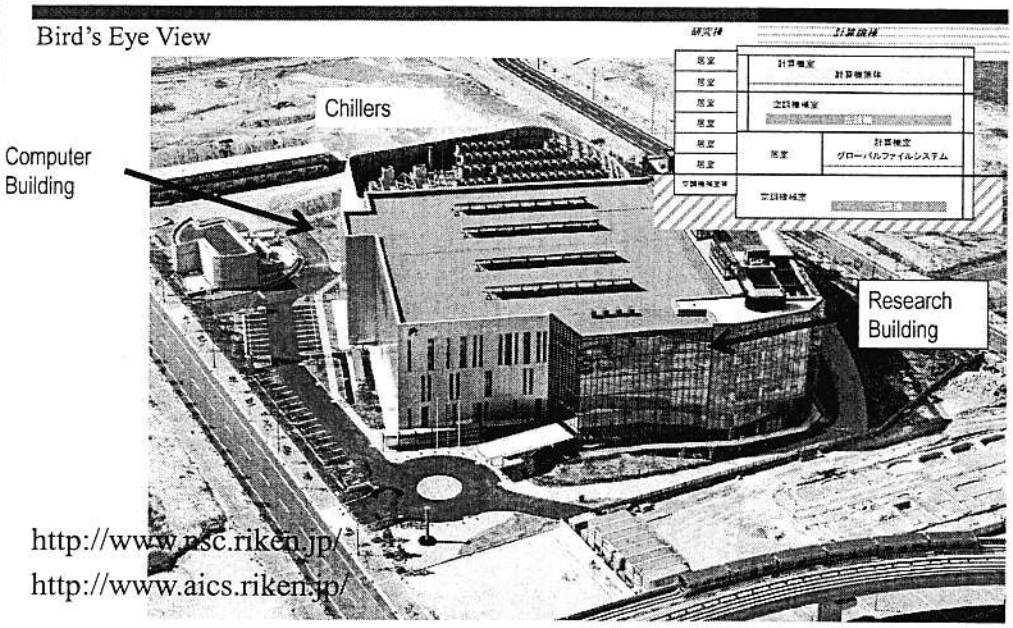
1

講義予定

- 10月18日　第 1回　イントロダクション
- 10月25日　第 2回　コンピュータの歴史、コンピュータの構成
- 11月 1日　第 3回　机上の計算機　演算、アドレッシング
- 11月 8日　第 4回　机上の計算機　サブルーチン
- 11月15日　第 5回　番外編：探索アルゴリズム（美添）
- 11月29日　第 6回　割り込み
- 11月22日　休み (駒場祭)
- 12月 6日　第 7回　パイプライン処理＆メモリ階層
- 12月13日　第 8回　仮想メモリ
- 12月20日　第 9回　プログラミング言語、ライブラリ、OS
- 12月24日　第10回　I/O (Disk, Network)
- 1月 7日　休講
- 1月17日　第11回　評価＆ベンチマーク＆まとめ
- 1月24日　試験

理研計算科学研究機構

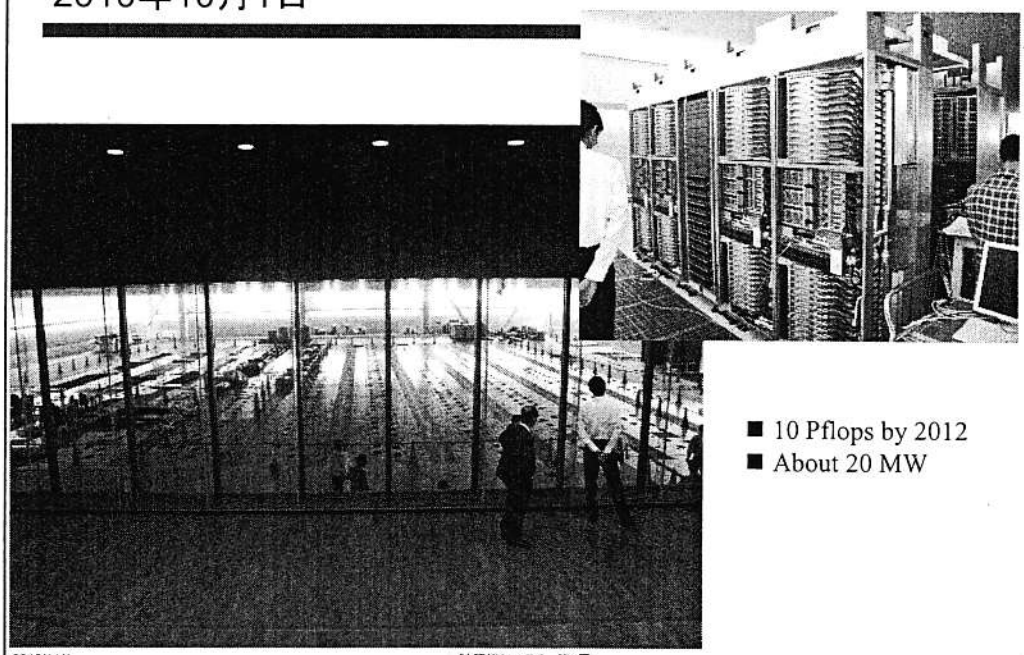
Bird's Eye View



http://www.nsc.riken.jp/

http://www.aics.riken.jp/

2010年10月1日



- 10 Pflops by 2012
- About 20 MW

コンピュータアーキテクチャ(復習)

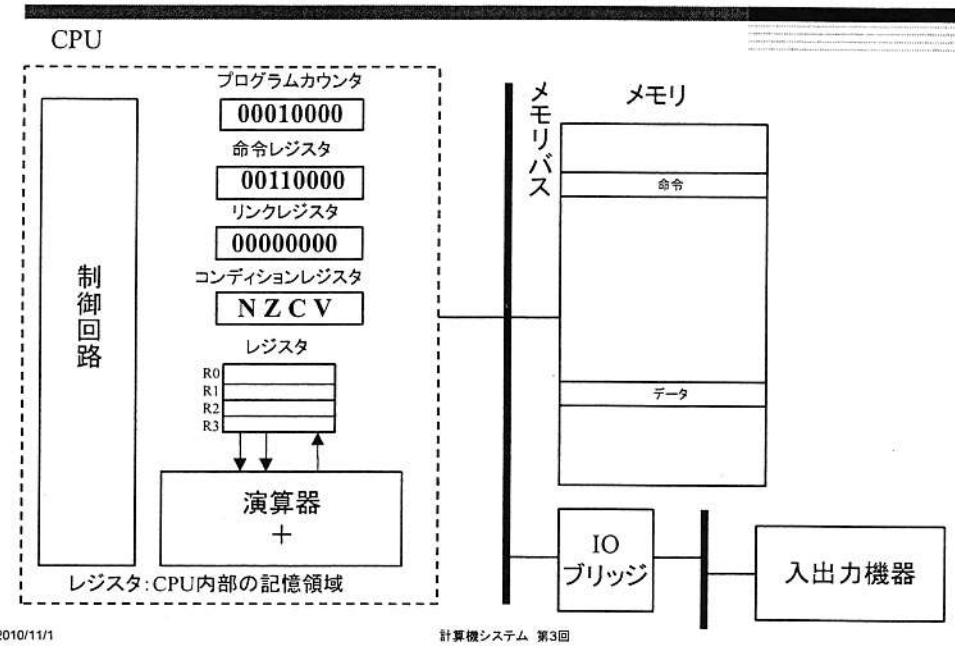
- G.Amdalh
  - Computer architecture is the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behaviour, as distinct from the organization of the data flows and controls the logic design, and the physical implementation.
  - IBM社360ファミリーコンピュータにコンパチビリティを与えた
- ソフトウェアから見たハードウェアの性質・属性
  - ユーザとハードウェアのインタフェース
    - 特に機械語プログラミング

機械語プログラミングに必要な情報(復習)

- ハードウェアが扱うデータ形式
  - 整数、符号なし整数、浮動小数点数、ピクセル値、文字、10進数、ストリング、アドレス、ポインタ、リスト、真偽値 等々
- 命令形式
  - 命令フォーマット、命令、オペランド
- メモリアドレスとメモリ操作
  - メモリアドレス空間、アドレシングモード、特殊アドレス(I/O等)
  - スタック、キューなどのメモリアクセス操作
- ユーザレジスタ、システムレジスタ、メモリ管理レジスタ
  - レジスタの意味、レジスタ内ビットの意味、メモリ管理方式
- メモリ管理
  - アドレス変換方式
- 割り込み
  - 割り込み要因と割り込み時操作
- プロセッサモードと保護機構
  - プロセス保護、メモリ保護、特権操作の定義

big endian/little endian

計算機の内部基本構成（復習）



浮動小数点（復習）

S	指数部（8ビット）	仮数部（23ビット）

$data = (-1)^S \times (1+\Sigma(fn/2^n)) \times 2^{E-127}$ （Σ:n=1-23）

1	01111110	11000000000000 ... 00

S=1, E=126

$(-1)^1 \times (1 + (1/2 + 1/2^2) + 0/2^3 + 0/2^4 + \ldots.) \times 2^{126-127}$

$= -(1 + 0.5 + 0.25) \times 2^{-1}$

$= -(1.75) \times 0.5 = -0.875$

浮動小数点と誤差(復習)

S	指数部(8ビット)	仮数部(23ビット)

$data = (-1)^S \times (1+\Sigma(fn/2^n)) \times 2^{E-127}$ (Σ:n=1-23)

- 丸め誤差
  - 有限桁数で表現することにより生じる誤差
  - 例:1.1 ➔ 0x3f8ccccc or 0x3f8ccccd

0	01111111	00011001100110011001100

0	01111111	00011001100110011001101

$(-1)^0 \times (1 + (1/2^4 + 1/2^5 + 1/2^8 + 1/2^9 + 1/2^{12} + 1/2^{13} + 1/2^{16} + 1/2^{17} + 1/2^{20} + 1/2^{21} \ldots.) \times 2^{127-127}$

$= 1 + 0.0625 + 0.0312 + 0.0039 + 0.0019 + \ldots = 1.0999999043$

$1.0999999043\ 1 + 1/2^{23} = 1.0999999043 + 0.0000001192 = 1.1000000235$

浮動小数点と誤差(復習)

S	指数部(8ビット)	仮数部(23ビット)

$data = (-1)^S \times (1+\Sigma(fn/2^n)) \times 2^{E-127}$ (Σ:n=1-23)

- 情報落ち
  - 絶対値の大きい数字と絶対値の小さい数字の加減算時に絶対値の小さい数字が無視される

$1.0 + 2^{-24} = 1.0 + 0.0000000596 = 1.0$

$1.0 + (2^{-24} + 2^{-24}) = 1.0 + 0.0000001192 = 1.0000001192$

1.0 = 0x3f800000, 1.0/(16*1024*1024) = 0x33800000

E=127

0	01111111	00000000000000000000000

$1 \times 2^0 + 1 \times 2^{-24}$
$= 1 \times 2^0 + (2^{-24} \times 2^0) = (1 + 2^{-24}) \times 2^0$

$1 \times 2^0 + 1 \times 2^{-23}$
$= 1 \times 2^0 + (2^{-23} \times 2^0) = (1 + 2^{-23}) \times 2^0$

E=103

0	01100111	00000000000000000000000

E=104

0	01101000	10000000000000000000000

浮動小数点と誤差

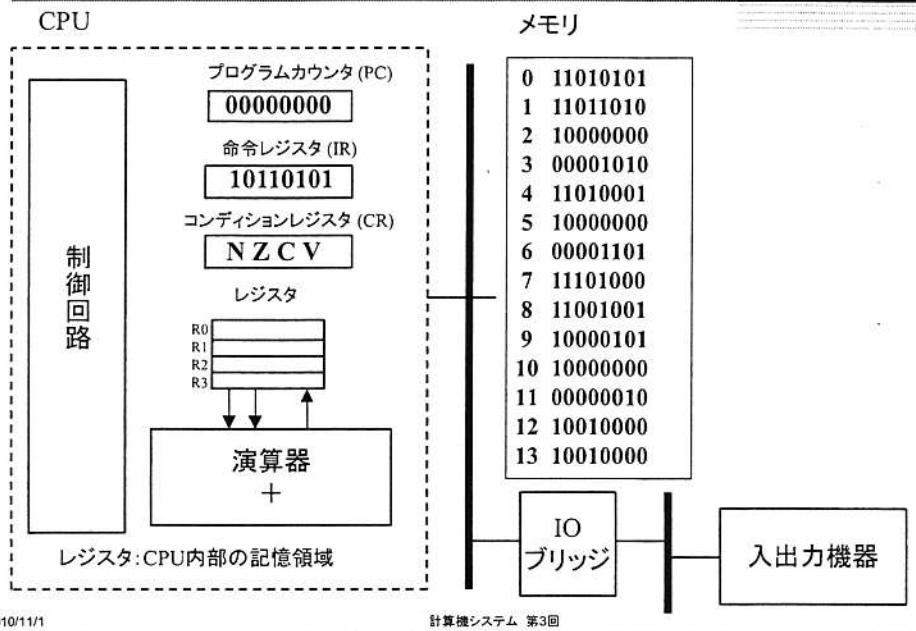
S	指数部（8ビット）	仮数部（23ビット）

$data = (-1)^S \times (1+\Sigma(fn/2^n)) \times 2^{E-127}$ $(\Sigma:n=1\text{-}23)$

- 桁落ち
  - 丸め誤差のある値で、ほぼ同じ値同士を減算したときに、有効桁数が落ちる

有効数字3ケタの値、
$0.123 - 0.122 = 0.001 = 0.1 \times 10^{-2}$
有効数字が1ケタになる

机上計算機の振る舞い



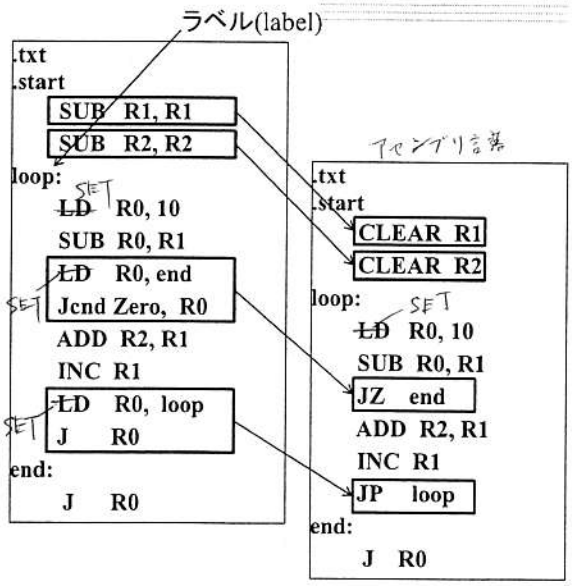
机上計算機　機械語

00000000	• NOP 何もせず、PCを+1する
00IIYYXX	• LD YY, II[XX] レジスタXXの内容に値II(0, -1, -2, -3)を足した値woメモリ番地とし、そのメモリの内容をレジスタYYに格納。PCを+1する
01IIYYXX	• ST YY, II[XX] レジスタYYの内容をレジスタXXの内容に値II(0, -1, -2, -3)を足した値をメモリ番地とたメモリに格納する。PCを+1する
100000XX	• SET XX, Immediate PC+1番地の内容をレジスタXXに格納し、PCを+2する
100001XX	• INC XX レジスタXXの内容を1足し、PCを+1する
100010XX	• DEC XX レジスタXXの内容を1引き PCを+1する
100011XX	• LDLNK XX LNKレジスタの内容をレジスタXXに格納
100100XX	• J XX レジスタXXの値にPCを変更
100101XX	• JLNK XX PC+1の値をLNKレジスタに格納し、レジスタXXの値にPCを変更する
100110XX	• IN XX キーボード入力値をレジスタXXに格納し、PCを+1する
100111XX	• OUT XX レジスタXXの内容をASCII文字として印字し、PCを+1する
101000XX	• LSHFT XX XX ← XX<<1 、PCを+1する
101001XX	• RSHFT XX XX ← XX >>1 、PCを+1する
101010XX	• AND XX XX ←XX and XX 、PCを+1する
1011YYXX	• MV YY, XX YY ← XX 、PCを+1する
1100YYXX	• ADD YY, XX YY ← YY + XX 、PCを+1する
1101YYXX	• SUB YY, XX YY ← YY - XX 、PCを+1する
1110CCXX	• Jcnd condition, XX コンディションCCが成立していれば、レジスタXの値にPCを変える
1111CCXX	• JLNKcnd condition, XX コンディションCCが成立していれば、PC+1の値をLNKレジスタに格納し、レジスタXの値にPCを変更する

機械語、アセンブリ言語

ニーモニック(mnemonic)
人が分かりやすいように機械語を記号表現したもの。LD, SUB, JZ

0	11010101	SUB R1, R1
1	11011010	SUB R2, R2
2	10000000	SET R0, 10
3	00001010	
4	11010001	SUB R0, R1
5	10000000	SET R0, 13
6	00001101	
7	11101000	Jcnd Zero, R0
8	11001001	ADD R2, R1
9	10000101	INC R1
10	10000000	SET R0, 2
11	00000010	
12	10010000	J R0
13	10010000	J R0



擬似命令、アセンブラ命令(pseudo instruction, pseudo operation)

- Pseudo instruction
  - 本来は存在しない機械語だが、アセンブラにより変換される

疑似命令 CLEAR R0 → アセンブラにより変換される（すなわち、レジスタ同士の減算） → 本当の命令 SUB R0, R0

- Pseudo operation
  - アセンブラに対する命令。機械語には変換されない。
  - 例
    - .text
    - .section

アセンブリ言語、アセンブラ

0	11010101
1	11011010
2	10000000
3	00001010
4	11010001
5	10000000
6	00001101
7	11101000
8	11001001
9	10000101
10	10000000
11	00000010
12	10010000
13	10010000

← アセンブリ言語で書かれたプログラムを機械語に翻訳

```
.txt
.start
     CLEAR R1
     CLEAR R2
loop:
     LD   R0, 10
     SUB  R0, R1
     JZ   end
     ADD  R2, R1
     INC  R1
     JP   loop
end:
     J    R0
```

高級言語 (High-level Programming Language)、コンパイラ

```
.txt
.start
      CLEAR  R1
      CLEAR  R2
loop:
      LD   R0, 10
      SUB  R0, R1
      JZ    end
      ADD  R2, R1
      INC  R1
      JP    loop
end:
      J    R0
```

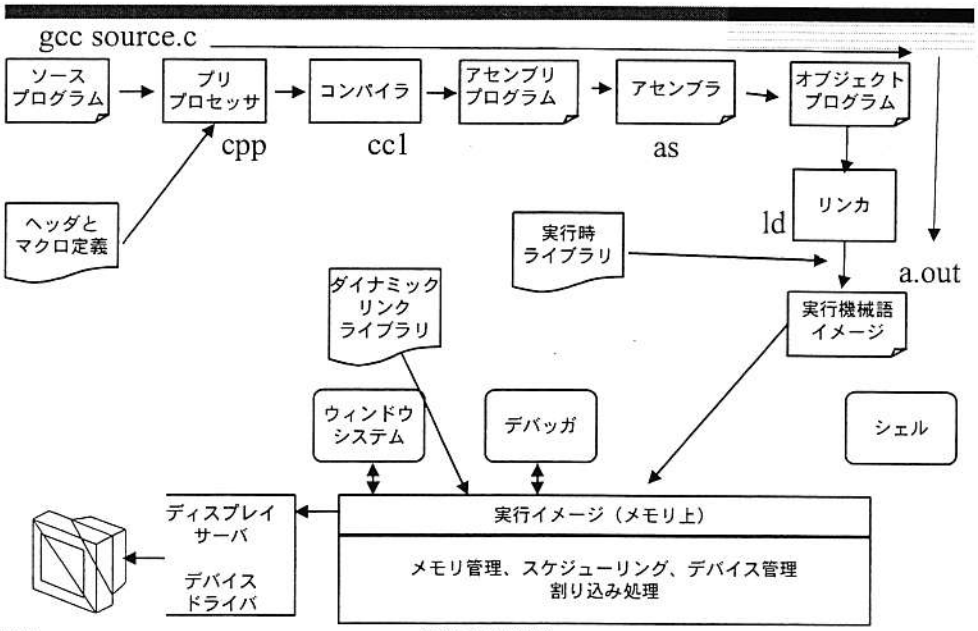
高級言語で書かれたプログラムをアセンブリ言語に翻訳

```
int i1 = 0;

for (i = 0; i < 10; i++) {
i1 = i1 + i;
}
```

高級言語(コンパイラ型):
アセンブリ言語のように計算機アーキテクチャに依存しないプログラミングするための言語。C/C++, Fortranなど。
Cは中級言語とも言われている。高級言語には、コンパイラ型とインタープリタ型がある。これらについては、第6回講義で説明予定

プログラムの実行



ルーチン、サブルーチン、手続き、ライブラリ

- プログラムのある仕事のまとまりをルーチン(routine)と呼ぶ。
- 繰り返し使えるプログラムをまとめたものをサーブルーチン (Subroutine)と呼ぶ。
- サブルーチンは関数(Function)、手続き(Procedure)とも呼ばれる。
- ライブラリとは、汎用性のある手続きをまとめて、いろいろなプログラムで使えるようにしたもの。

機械語を読む

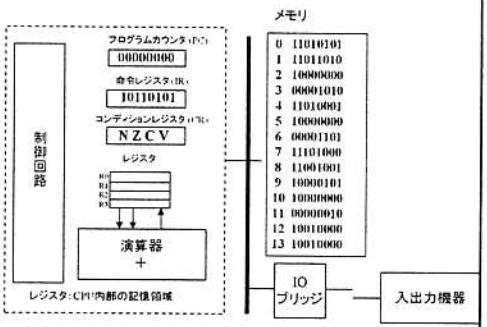
0	11010101	SUB R1, R1
1	11011010	SUB R2, R2
2	10000000	SET R0, 10
3	00001010	
4	11010001	SUB R0, R1
5	10000000	SET R0, 13
6	00001101	
7	11101000	Jcnd Zero, R0
8	11001001	ADD R2, R1
9	10000101	INC R1
10	10000000	SET R0, 2
11	00000010	
12	10010000	J R0
13	10010000	J R0

Condition Register

- 演算結果の状態を表現する
  - 整数演算、浮動小数点演算
- 演算しない命令の実行後は、コンディションレジスタは変化しない
  - レジスタロード・ストア、分岐、、

コンディションレジスタ

N	Z	C	V



コンディションコード

- N – Negative
  - 計算結果が2の補数表現で負の時にセット
- Z – Zero
  - 演算結果がゼロの時にセット
- V – Overflow
  - 計算結果が2の補数表現で桁に収まらなかったときにセット
- C – Carry (Borrow)
  - 計算結果、桁上げ(Carry)した、あるいは桁下げ(Borrow)が行われたとき

コンディションコード

- V – Overflow
  - 計算結果が2の補数表現において桁に収まらなかったときにセット
    - 加算命令の場合
      - 正数と正数の加算で，最上位ビットが1になり負数になってしまう。
      - 負数と負数の加算で，最上位 ビットが0になり正数になってしまう。
    - 減算命令の場合
      - 例を考えてみましょう。

コンディションコード

- C – Carry (Borrow)
  - 計算結果、桁上げ(Carry)した、あるいは桁下げ(Borrow)が行われたとき

```
  10000000          (0)00000000
+ 10000000        -  10000000
 (1)00000000        10000000
```

  - キャリーフラグを含めた演算がある
    - ADDX
      - add ~~with carry~~ — Sparcの場合
    - SUBX
      - subtract with carry

コンディションコード

- C – Carry (Borrow)
  - シフト命令等で最上位ビットあるいは最下位ビットがCarryに回る

SLL: Shift Left Logical
SRL: Shift Right Logical

加減算および乗除算(一般論)

- 加減算命令
  - 符号付、符号なしの区別はない
    - コンディションコードの扱いの違いはある
- 乗除算
  - 符号付、符号なしの区別がある

コンディションコードと分岐命令

- 机上計算機の場合

100100XX
- J XX
  - レジスタXXの値にPCを変更

100101XX
- JLNK XX
  - PC+1の値をLNKレジスタに格納し、レジスタXXの値にPCを変更する

1110CCXX
- Jcnd condition, XX
  - コンディションCCが成立していれば、レジスタXの値にPCを変える

1111CCXX
- JLNKcnd condition, XX
  - コンディションCCが成立していれば、PC+1の値をLNKレジスタに格納し、レジスタXの値にPCを変更する

コンディションレジスタ

N	Z	C	V

00	Vフラグが立っていれば
01	Cフラグが立っていれば
10	Zフラグが立っていれば
11	Nフラグが立っていれば

Operation Code & Operand &Addressing Mode

- Operation Code
  - 演算命令
- Operand
  - 演算命令に使用するもの
  - 指定方法
    - レジスタ
    - 即値
    - ディスプレースメント
    - レジスタ間接
    - インデックス修飾
    - 直接
    - メモリ間接
    - 自動インクリメント
    - 自動ディクリメント
    - スケール付きインデックス修飾

アドレシングモード

アドレシング・モード	命令例	意味	使用する場合
レジスタ	Add R4, R3	R4 ← R4＋R3	値がレジスタに存在する時
即値またはリテラル	Add R4, #3	R4 ← R4＋3	定数として使用される．あるマシンでは，リテラルと即値は別々のアドレシング・モードとなっている．
ディスプレースメント（ベース相対）	Add R4, 100 (R1)	R4 ← R4＋M [100＋R1]	局所変数にアクセスする時
レジスタ間接	Add R4, (R1)	R4 ← R4＋M [R1]	ポインタまたはアドレス計算結果によってアクセスする時
インデックス修飾	Add R3,(R1＋R2)	R3 ← R3＋M [R1＋R2]	配列のアドレシングに有用となることが多い．R1 は配列のベース，R2 はインデックス量．
直接または絶対	Add R1,(1001)	R1 ← R1＋M [1001]	静的データにアクセスする際に有用となることが多い．アドレス定数は非常に長いものが要求される．
メモリ間接	Add R1,@(R3)	R1 ← R1＋M [M [R3]]	R3 がポインタ p のアドレスを指すとすると，このモードはポインタ p の指すメモリの内容・p にアクセスする．
自動インクリメント	Add R1,(R2)+	R1 ← R1＋M [R2] R2 ← R2＋d	ループを用いて配列を 1 つ 1 つ処理する際に有用である．R2 は配列の開始アドレスを指す．各々の参照ごとに R2 には要素サイズ d を加算する．
自動デクリメント	Add R1,−(R2)	R2 ← R2−d R1 ← R1＋M [R2]	自動インクリメントと同様．これら 2 つのモードはスタックのプッシュとポップにも利用できる．
スケール付きインデックス修飾	Add R1, 100 (R2)[R3]	R1 ← R1＋M[100＋R2＋R3＊d]	配列のインデックス付けに使用される．あるマシンではベース・アドレシング・モードのどれにも適用できる．