2008－ARC－179 (32)
2008／8／7

COMETⅡの桁上げと桁あふれに関する提案

加藤 肇彦

情報処理技術者試験用仮想プロセッサ COMETⅡの桁上げと桁あふれに関する提案を示す．

算術左シフト命令の桁上げの定義については，古典的な符号保存シフトに替えて，現在マイクロコンピュータで普及している全ビットシフトを採用することによって，算術加算命令との整合性が達成されることを示す．算術左シフトによる桁あふれについてはその定義の矛盾を反例によって示し，現在広く合意されている定義に合わせることを提案する．算術加減算による桁あふれの判定については，加減算を統一的に検討し，内部２進で動作する処理機構での現実的な判定基準に準拠することを提案する．

A Proposal for the Carry and Overflow of the COMETⅡ Processor

HATSUHIKO KATO

A proposal will be made for the carry and overflow of the COMETⅡ virtual processor used for the Information Processing Engineers' Examination.

As for the definition of the overflow in the arithmetic shift left instruction, the classical sign-preserving shift will be substituted with the contemporary whole bits shift. As the consequence, the consistency with the arithmetic addition instruction will be acquired. As for the overflow resulting from the arithmetic shift left instruction, the inconsistency in the definition of the overflow bit will be pointed out with some counterexamples. The compliance to the widely accepted definition of overflow will be proposed. As for the overflow due to the arithmetic addition and subtraction, a unified study will be made for both operations and a practical criterion will be proposed for the processing unit internally operating in binary principle.

1．はじめに

１９６９年に情報処理技術者試験（以下試験）の制度が発足して以来，プログラム言語の選択肢として仮想プロセッサ COMET と，そのアセンブラ言語である CASL が出題されてきた [1]．これらはマイクロコンピュータの普及を反映して，2000 年度より仕様に変更を加えられ，それぞれ COMETⅡおよび CASLⅡと改名された [2, 3]．現在これらの仮想プロセッサとアセンブラ言語は，単に試験の想定計算機としての位置付けを超えて，アーキテクチャとアセンブラ言語の教育の標準モデルとしても広く使用されている．さらに 2002 年度からは諸外国との相互認証の協定を結び，協力体制の構築が進められている．このような状況から，COMETⅡおよび CASLⅡの仕様は本邦ならびに近隣諸国の情報処理教育の共有財産といっても過言ではない．従ってこれらの仕様は普遍性をもち，かつ情報処理の最新動向を反映したものでなければならない．

2000 年度の改訂にも拘らず，COMETⅡの仕様には今なお試験発足当時主流であった汎用大型計算機（以下大型機）の方式を踏襲した部分が見受けられる．大型機のプログラム

表１．　COMETⅡの算術左シフト命令と加減算命令の結果比較

（ａ）正数Ａ＝２４０２６＝（０１０１１１０１１１０１１０１０）$_{2C}$を２倍した結果

被乗数　Ａに対する操作	OF フラグ	結　果
SLA 命令による２倍　2A	1	００１１１０１１１０１１０１００
ADDA 命令による加算　A+A	1	１０１１１０１１１０１１０１００
SUBA 命令による減算　A−(−A)	1	１０１１１０１１１０１１０１００

（ｂ）負数－Ａ＝－２４０２６＝（１０１０００１０００１００１１０）$_{2C}$を２倍した結果

被乗数　－Ａに対する操作	OF フラグ	結　果
SLA 命令による２倍　－2A	0*	１１０００１０００１００１１００
ADDA 命令による加算　－A+(−A)	1	０１０００１０００１００１１００
SUBA 命令による減算　－A−A	1	０１０００１０００１００１１００

SF フラグは MSB に等しい　＊：桁あふれが発生するにもかかわらずＯＦには０が設定される

作成には高水準言語が常用される現在，アセンブラが使用される局面はマイクロコンピュータの組み込み応用に多く見られるため，COMETⅡの仕様は大型機ではなくマイクロコンピュータを想定することが望ましい．さらに，COMETⅡの仕様では桁あふれ表示ビットの機能に明確な矛盾が含まれている．本発表では，特に COMETⅡの桁上げと桁あふれの仕様に注目して，その問題点を指摘し，情報処理の現状に即した変更を提案する．

２．算術左シフト命令の仕様

COMETⅡでは実効アドレスを指標即値オペランドで指定可能であり，シフト命令では実効アドレスによってシフトすべきビット数が指定できる．算術左シフト命令 SLA においてシフトビット数を１と指定することは２の補数表示のオペランドに対する２倍の操作を意味する．２倍操作は算術加算命令 ADDA で同じ値のオペランド同士を加算しても可能であり，いずれの方法で実行しても結果の値も桁あふれフラグも一致することが望ましい．現行 COMETⅡの算術シフトの仕様では「符号を除き（ｒ）を実効アドレスで指定したビット数だけ左又は右にシフトする」と定義されている．この定義によれば桁あふれがない場合は SLA と ADDA の実行結果は一致するが，桁あふれをともなう場合は表１に示すようにこれら２種類の命令で異なる結果が得られる．SLA では符号ビットを保存して数値ビットのみ左シフトしているので，例えば 16 ビット２の補数表示で 2倍可能な最大数である 16383=(0011111111111111)$_{2C}$ を上回る値 24026＝(0101110111011010)$_{2C}$ を 2 倍しようとすると(0011101110110100)$_{2C}$ となり，符号は０のままで変化しない．一方 ADDA による結果は(1011101110110100)$_{2C}$ となり，数値ビットは同じであるが符号は１に変化して SLA とは異なる結果が得られる．同様に 2 倍可能な数の下限である－16384=(1100000000000000)$_{2C}$ を下回る－24026＝(1010001000100110)$_{2C}$ を SLA で２倍しようとすると(1100010001001100)$_{2C}$ となり，符号は１のまま変化しないで保存される．しかもこの場合桁あふれが発生するにも拘らずフラグレジスタ FR 内の桁あふれフラグ OF は０となり，不正の結果を示す．ADDA では結果は(0100010001001100)$_{2C}$ となり，数値ビットは同じであるが符号は０に変化して OF は１となり，SLA とは異なる結果を示す．

これらの例に示した SLA で符号ビットを保存する方式は符号と絶対値表示の算術左シフトに起源が見られ，2 の補数表示が採用されてからも 1960 年代の大型機が踏襲してきたものである [4]．しかし現在のミニコンピュータおよびマイクロコンピュータの処理機構の算術左シフトでは，すべて符号と数値を含めた全ビット左シフトの方式に移行している [5], [6]．（註：JIS の情報処理用語では情報処理システムの内，慣用的に CPU と呼ばれている部分を「処理機構」，処理機構と主記憶を組み合わせたものを「処理装置」と呼んで区別

表２．　COMETⅡの算術左シフト後の桁あふれフラグＯＦ

１０進相当値	算術左シフト前	算術左シフト後	OF フラグとその正否
16384 ～ 32767	０１×××…×	０×××…×０ 桁あふれ	１　正常
０ ～ 16383	００×××…×	０×××…×０ 正常	０　正常
−16384 ～ −1	１１×××…×	１×××…×０ 正常	１　不正
−32768 ～ −16385	１０×××…×	１×××…×０ 桁あふれ	０　不正

している [7]．本稿でもこの定義に準拠する）さらに大半の機種では算術左シフトと論理左シフトを統一して，左シフトを１種類の命令で実行するようになっている．このような現状に合せ，COMETⅡでも左シフトに関しては算術も論理も共通して，全ビット左シフトに切り替えることが望ましい．

符号保存左シフトの弊害は数値計算において具体的に現れる．例えばCOMETⅡのように乗除算命令をハードウェアで用意していない低位の処理機構では，その機能を数値計算プログラムとして実行する．除算を引放し法によって実行する場合，部分剰余を算術左シフトにより２倍した後除数を加減する操作を反復する．この操作では算術左シフトによって桁あふれが発生することがあるが，全ビット左シフトでは有効な情報が保存されているため，次の操作で桁あふれが打ち消されて正確に続行できる．しかし符号保存左シフトではシフト前の数値ビットの最上位の情報が失われ，以降の操作が無意味になって除算は実行できない．この問題は除算だけでなく算術左シフトを含むすべての数値計算においても発生する．

３．算術左シフト後の桁あふれフラグ

前章で大きな絶対値をもつ負数をSLAで２倍しようとした場合のOFフラグの不正結果の一例を示したが，ここでSLAにおいてOFフラグが正しく得られる場合と不正になる場合を統一的に考察する．COMETⅡの仕様ではOFフラグは「レジスタから最後に送り出されたビットの値が設定される」となっているが，この定義が過去に実在した処理機構でも現存する処理機構でも採用された例はなく，実務に使用される処理機構を想定したプログラム作成能力を検定する試験の趣旨に一致しているとはいえない．

表２にCOMETⅡによるSLA実行前後の結果を算術左シフト前の上位２ビットの４種類の組み合わせによって分類して示す．２の補数表示では上位２ビットが00または11である場合は２倍可能な範囲に収まるが，10進相当値が16384以上であれば２倍した値は$2^{15}-1=32767$を上回り，−16385以下の場合は２倍した値は$-2^{15}=-32768$を下回るため桁あふれを起こして２倍できない．上記のOFフラグの定義によれば，符号保存算術左シフト実行後は実行前の上から２ビット目の値が設定されることになり，表２に示すとおり負数を算術左シフトした場合は桁あふれの有無にかかわらずOFフラグには逆の結果が設定されてしまう．全ビット左シフトを採用している現在のミニコンピュータやマイクロコンピュータはもちろん，符号保存左シフトを採用している大型機も含めて，実在の処理機構の仕様ではデータのビット数に関係なく，すべて「桁あふれビットには，算術左シフト前の上位２ビットが00または11である値をシフトした場合０が設定され，上位２ビットが01または10である値をシフトした場合１が設定される」と定義されている．これは言い換えれば「上位２ビットにおける桁上げが発生しないか，数値ビットから符号ビットへ，そして符号ビットからさらに上位へ２回発生する場合は正常，桁上げがいずれか１回だけの場合は桁あふれである」と表現でき，次章で述べる加減算実行後のOFフラグの設定とも整合する．COMETⅡの仕様においても，この定義に準拠することが望ましい．算術シフト命令はシフト命令に分類されると同時に，その機能は算術加減算命令との整合性を保たなければならない．OFフラグを最上位ビットからの桁上げ（論理桁あふれ）Cと，算術桁

表 3.　2の補数表示の加算結果例

被 加 数 A	加　数 B	和 A＋B	桁上げ回数	図 1 上の領域
0101001101101011	0011100100110101	1000110010100000	1回*	① 桁あふれ
0011100010101100	0001111011010110	0101011110000010	0回	② 正常
0111010101011101	1100111000101110	0100001110001011	2回	③ 正常
1100111000101110	0111010101011101	0100001110001011	2回	③' 正常
1011100110110101	1110101111001001	1010010101111110	2回	④ 正常
0001101001110011	1001011000110110	1011000010101001	0回	⑤ 正常
1001011000110110	0001101001110011	1011000010101001	0回	⑤' 正常
1001011101100111	1010101001010010	0100000110111001	I回**	⑥ 桁あふれ

＊：1回は数値桁→符号桁の桁上げ　　＊＊：I回は符号桁→上位の桁上げ

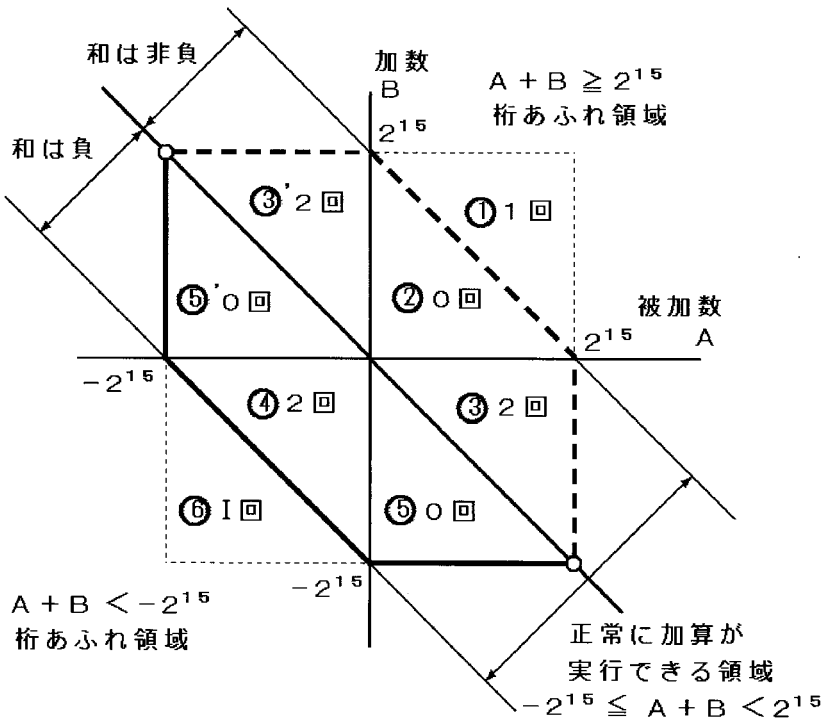

図1．2の補数表示の加算結果（桁上げの回数と加算可否の関係）

あふれVのフラグに分けて区別し，左シフトの論理・算術の区別を撤廃するという，現在一般的に用いられている方式に準拠することも考えられる．この場合は桁上げ分岐命令JOCの新設が必要である．

4．算術加減算後の桁あふれフラグ

COMETⅡの仕様によれば算術加減算実行後のOFフラグは「算術演算命令の場合は，演算結果が−32768～32767に収まらなくなったとき1になり，それ以外のとき0になる」と定義されていて，これは前章の提案と表2に示す結果と整合する．しかし内部2進で動作する実在のすべての処理機構では演算結果を直接10進数と比較することはできないため，算術加算の場合は前章に示した上位2ビ

表 4.　2の補数表示の減算結果例

被 減 数 A	減　数 B	差 A−B	借り回数	図2上の領域
0101001101101011	1100011011001011	1001110010100000	1回*	① 桁あふれ
0011100010101100	1110000100101010	0101011110000010	2回	② 正常
0111010101011101	0011000111010010	0100001110001011	0回	③ 正常
1100111000101110	1000101010100011	0100001110001011	0回	③' 正常
1011100110110101	0001010000110111	1010010101111110	0回	④ 正常
0001101001110011	0110100111001010	1011000010101001	2回	⑤ 正常
1001011000110110	1110010110001101	1011000010101001	2回	⑤' 正常
1001011101100111	0101010110101110	0100000110111001	I回**	⑥ 桁あふれ

*：1回は上位→符号桁の借り　　**：I回は符号桁→数値桁の借り

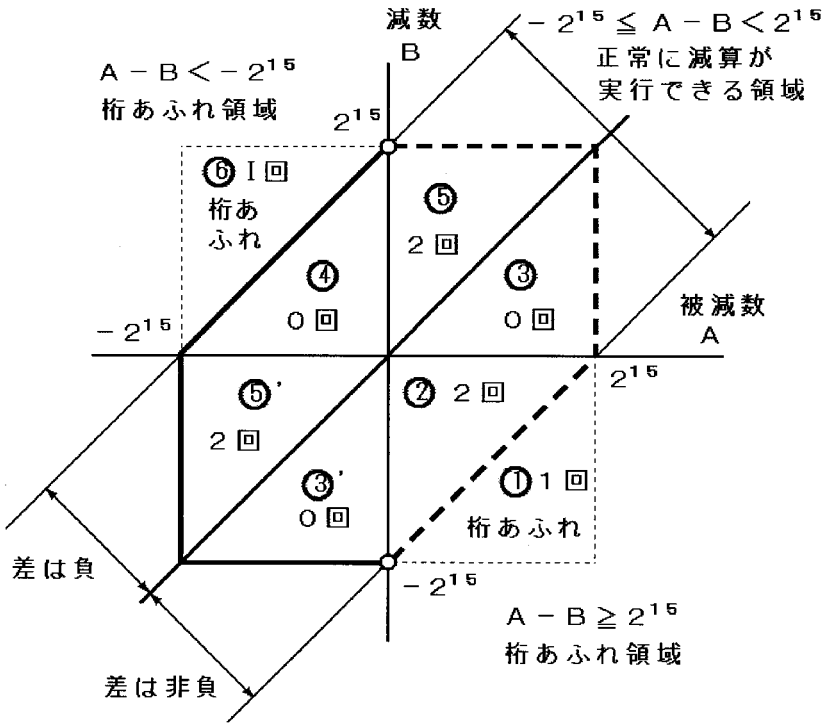

図2．2の補数表示の減算結果（借りの回数と減算可否の関係）

ットの桁上げの回数によって桁あふれの有無を判定している．また，アーキテクチャの教科書や参考書にも広くこの判定法が掲載され，アセンブラによるプログラミングの担当者の間でも日常の実務でこの判定法が流布している[8)-11)]．表3と図1は2の補数表示の被加数と加数に対する加算結果と桁あふれの関係を，桁上げ回数により分類して示している．SLAによる2倍操作は被加数と加数が等しい場合の加算とみなすことができる．表2の算術左シフト前の上位2ビットが11である場合は図1では絶対値の小さな負数が正常に2倍される領域④に相当し，上位2ビットが10である場合は絶対値の大きな負数を2倍すると桁あ

ふれを起こす領域⑥に相当する．

算術減算は通常減数の符号を反転して加算に帰着させることが多いが，減数を直接被減数から減ずることも可能である．その場合も算術加算における桁上げと同様，上位2ビットでの借りの発生回数によって桁あふれの有無を判定できる．表4と図2は2の補数表示の減数と被減数に対する減算結果と桁あふれの関係を，借りの発生回数により分類して示している．これは「上位2ビットにおける借りが発生しないか，符号ビットから数値ビットへ，そして上位から符号ビットへ2回発生する場合は正常，借りがいずれか1回だけの場合は桁あふれである」と表現される．

算術加減算による桁あふれの判定条件を統一すれば，「上位2ビットにおける桁上げまたは借りが発生しないか，2回発生する場合は正常，桁上げまたは借りがいずれか1回のみの場合は桁あふれである」と表現できる．さらにこの判定基準は3章に示したとおり，算術左シフトによる桁あふれの定義とも整合する．実在の処理機構でも上位2ビットにおける桁上げまたは借りの異同を exclusive OR（排他的論理和）回路によって検出し，桁あふれを判定している．

5．まとめ

2章から4章に示した算術左シフトならびに算術加減算に関する提案は，以下に示す2項目に集約することができる．

（1）算術左シフトは符号ビットと数値ビットを含めて，全ビットを実効アドレスで指定されたビット数だけ左に桁送りする．（推奨）

（2）算術左シフトならびに算術加減算の結果，上位2ビットにおいて発生する桁上げまたは借りが0回または2回であれば，桁あふれフラグOFに0が設定され，1回であれば桁あふれフラグOFに1が設定される．（必須）

これらの項目の内（1）は定義の問題であり，しかも大型機にはソフトウェア互換性維持のためこの定義によらない機種も残存しているが，現行の圧倒的多数の処理機構は（1）に準拠しているため，COMETⅡもこちらに整合させることが望ましい．また，（2）は2の補数表示の加減算原理そのものであり，現行の COMETⅡの仕様の矛盾と不整合を回避して，正確な演算結果を得るための必須の遵守事項である．計算機アーキテクチャは個々の命令やレジスタの集合体であるだけでなく，それらすべてが関連して機能する有機体と捉えるべきである．本提案が情報処理技術者試験の改善に寄与すれば光栄である．

参 考 文 献

1) (財)日本情報処理開発協会 情報処理技術者試験センター編：情報処理技術者試験案内書・願書(1999).
2) 独立行政法人情報処理推進機構 情報処理技術者試験センター編：情報処理技術者試験案内書・願書(2008).
3) 独立行政法人情報処理推進機構情報処理技術者試験センターウェブページ，www.jitec.jp，(2008).
4) [390 principle of operation]，www.s390.ibm.com/bkserv/hw/, IBM Corporation，(2004).
5) 喜田祐三，萩原吉宗，岩崎一彦：MC68000マイクロコンピュータ，pp.10-12，丸善（東京），(1983).
6) IA-32 Architecture Software Developer's Manuals，Vol.2A, pp.4-235-240, Intel Corporation，(2005).
7) 日本規格協会編：JISハンドブック64情報基本用語・データコード編(2005).
8) 中沢喜三郎：計算機アーキテクチャと構成方式，pp.246-249，朝倉書店，東京(1995).
9) 富田眞治：コンピュータアーキテクチャ第2版，pp.131-135，丸善㈱，東京(2000).
10) 馬場敬信：コンピュータアーキテクチャ改訂第2版，pp.145-147，オーム社，東京(2005).
11) 萩原宏，黒住祥祐：現代電子計算機ハードウェア改訂第2版，pp.77-80，オーム社，東京(1998).