# Amber Trace Format Specification v1.5

## TT6 trace format

TT6 records consist of entries that are one, two, or three longwords (32-bits) in length. Each longword of data is stored in big-endian format.

Record 1 is an INITIAL_PC record consisting of one longword. It is the PC (address) of the first instruction in the trace.

Records 2-N are instruction trace records of the following types:

COMPUTE

**longword 1**: instruction opcode (all instructions that do not fall under the FLOW_ALTERING, MEMORY and MEMORY_EXTENDED classes)

FLOW_ALTERING

**longword 1**: instruction opcode (change of flow class; branches, rfi, sc)
**longword 2**: address of next instruction to be executed

MEMORY

**longword 1**: instruction opcode (load/store class, where the size of the operand can be determined from the opcode; most load/stores, cache ops, etc)
**longword 2**: address of data operand

MEMORY_EXTENDED

**longword 1**: instruction opcode (load/store class, where the size of the operand or other parameter CANNOT be determined from the opcode. Currently includes load/store string indexed (third longword is contents of XER register) and AltiVec DST (third longword is contents of rB which indicated stream parameters) instructions.
**longword 2**: address of data operand
**longword 3**: byte count of data operand size

## TT6E trace format

TT6E is different from the original TT6 format in the following ways:

- effective addresses for **DCB**x instructions are emitted
- synthetic instructions are emitted when TWI instructions are encountered, simulating a return from a trap:
    1. b 0x00000700
    2. rfi

## FULL trace format

FULL trace format records the contents of all user state (integer, floating point and vector registers as well as other state such as the CR and XER registers) for every instruction executed. The first longword of a FULL trace is the magic cookie 0xA1118E17. The magic cookie is followed by N trace record entries:

**longword 1**: instruction opcode

**struct ppc_thread_state**: all integer registers as well as the PC (srr0) and MSR (srr1), CR, XER, LR, CTR and VRSAVE registers

**struct ppc_float_state**: all floating point registers as well as the FPSCR register

**struct ppc_vector_state**: all vector registers as well as VSCR register.

All of the above structures are defined in /usr/include/mach/ppc/thread_status.h.

## Thread Synchronization Records

Amber is able to instrument traces with thread synchronization events. Because all MacOS X synchronization functions rely on POSIX thread (pthread) primitives, amber is able to provide a record of thread synchronization by tracking calls to the following pthread functions in the traced executable:

```
pthread_mutex_lock()
pthread_mutex_unlock()
pthread_mutex_trylock()

pthread_cond_broadcast()
pthread_cond_signal()
pthread_cond_wait()
```

Amber outputs one trace file per thread. Trace files are instrumented with synchronization information when the -p option is used.

In order to embed the information in the trace file, TT6 escape records are used:

```
typedef struct {
    uint32_t escape : 6;     /*  0- 5 escape record indicator - always zero */
    uint32_t code : 10;     /* 10-15 code:  0x00 = segment register
                            0x01 = optional data address for next instruction
                            0x02 = condition register value
                            0x03 = real address of branch target for next instruction
                            0x04 = real data address for next instruction
                            0x05 = real address of next instruction
                            0x20-0x3F = thread synchronization */
    uint32_t count : 16;     /* 16-32 number of words following escape record */
} tt6_escape_record_fields_t;

typedef union {
    tt6_escape_record_fields_t field;
    uint32_t value;
} tt6_escape_record_t;
```

There are currently four types of thread synchronization escape records (the escape record range 0x20-0x3F is reserved for future use):

```
typedef enum {
    SYNC_SIGNAL             = 0x20,
    SYNC_BROADCAST_SIGNAL    = 0x21,
    SYNC_WAIT           = 0x30,
    SYNC_TRY_WAIT       = 0x31,
} sync_type_t;
```

```
// signal escape record:
// [Word 0] 0 : 0x20 : 2
// [Word 1] sync point address
// [Word 2] sync point count

// broadcast signal escape record:
// [Word 0] 0 : 0x21 : 2
// [Word 1] sync point address
// [Word 2] sync point count

// wait escape record:
// [Word 0] 0 : 0x30 : 2
// [Word 1] sync point address
// [Word 2] sync point count

// try wait (succeeded) escape record:
// [Word 0] 0 : 0x31 : 2
// [Word 1] sync point address
// [Word 2] sync point count
```

Each time one of the traced synchronization SYNC_SIGNAL or SYNC_BROADCAST_SIGNAL functions is encountered, the count associated with the mutex (or condition variable) is incremented.

Other notes:

Escape record for signal synchronization functions (pthread_mutex_unlock, pthread_cond_signal, pthread_cond_broadcast) are inserted at the beginning of the function call, before the instructions for the function appear in the trace. Wait function escape records appear in the trace after the corresponding synchronization function (pthread_mutex_lock, pthread_mutex_trylock, pthread_cond_wait) has completed, and therefore the trace after the traced instructions for that function.

SYNC_TRY_WAIT records only appear in the trace when the try was successful.

Internal calls to synchronization functions (for example, pthread_cond_wait() calls pthread_mutex_lock()) do not appear in the trace. Only the top-level synchronization function will appear (pthread_cond_wait()).

## TT6 Instruction Class Macros

```
#define tt6_isFlowAltering(majorOp, minorOp) ( \
(majorOp==18) ||                    /* b, ba, bl, bla       */    \
(majorOp==16) ||                    /* bc, bca, bcl, bcla   */    \
(majorOp==19 && minorOp==528) ||    /* bcctr, bcctrl        */    \
(majorOp==19 && minorOp==16)  ||    /* bclr, bclrl          */    \
(majorOp==19 && minorOp==50)  ||    /* rfi                  */    \
(majorOp==17) ||                    /* sc                   */    \
(majorOp==19 && minorOp==18)        /* PPC64: rfid          */    \
)

#define tt6_isMemory (majorOp, minorOp) ( \
(majorOp==34) ||  /* lbz      */          \
(majorOp==35) ||  /* lbzu     */          \
(majorOp==50) ||  /* lfd      */          \
(majorOp==51) ||  /* lfdu     */          \
(majorOp==48) ||  /* lfs      */          \
(majorOp==49) ||  /* lfsu     */          \
(majorOp==42) ||  /* lha      */          \
(majorOp==43) ||  /* lhau     */          \
(majorOp==40) ||  /* lhz      */          \
(majorOp==41) ||  /* lhzu     */          \
(majorOp==46) ||  /* lmw      */          \
(majorOp==32) ||  /* lwz      */          \
(majorOp==33) ||  /* lwzu     */          \
(majorOp==31 && minorOp==597) ||    /* lswi     */          \
(majorOp==31 && minorOp==119) ||    /* lbzux    */          \
(majorOp==31 && minorOp==87)  ||    /* lbzx     */          \
(majorOp==31 && minorOp==631) ||    /* lfdux    */          \
(majorOp==31 && minorOp==599) ||    /* lfdx     */          \
(majorOp==31 && minorOp==567) ||    /* lfsux    */          \
(majorOp==31 && minorOp==535) ||    /* lfsx     */          \
(majorOp==31 && minorOp==375) ||    /* lhaux    */          \
(majorOp==31 && minorOp==343) ||    /* lhax     */          \
(majorOp==31 && minorOp==790) ||    /* lhbrx    */          \
(majorOp==31 && minorOp==311) ||    /* lhzux    */          \
(majorOp==31 && minorOp==279) ||    /* lhzx     */          \
(majorOp==31 && minorOp==597) ||    /* lswi     */          \
(majorOp==31 && minorOp==20)  ||    /* lwarx    */          \
(majorOp==31 && minorOp==534) ||    /* lwbrx    */          \
(majorOp==31 && minorOp==55)  ||    /* lwzux    */          \
(majorOp==31 && minorOp==23)  ||    /* lwzx     */          \
(majorOp==31 && minorOp==7)   ||    /* lvebx    */          \
(majorOp==31 && minorOp==39)  ||    /* lvehx    */          \
(majorOp==31 && minorOp==71)  ||    /* lvewx    */          \
(majorOp==31 && minorOp==103) ||    /* lvx      */          \
(majorOp==31 && minorOp==359) ||    /* lvxl     */          \
(majorOp==31 && minorOp==310) ||    /* eciwx    */          \
(majorOp==31 && minorOp==438) ||    /* ecowx    */          \
(majorOp==38) ||  /* stb      */    \
(majorOp==39) ||  /* stbu     */    \
(majorOp==54) ||  /* stfd     */    \
(majorOp==55) ||  /* stfdu    */    \
(majorOp==52) ||  /* stfs     */    \
(majorOp==53) ||  /* stfsu    */    \
(majorOp==44) ||  /* sth      */    \
(majorOp==45) ||  /* sthu     */    \
(majorOp==47) ||  /* stmw     */      \
(majorOp==36) ||  /* stw      */      \
(majorOp==37) ||  /* stwu     */      \
```

```
(majorOp==31 && minorOp==247) ||    /* stbux    */    \
(majorOp==31 && minorOp==215) ||    /* stbx     */    \
(majorOp==31 && minorOp==759) ||    /* stfdux   */    \
(majorOp==31 && minorOp==727) ||    /* stfdx    */    \
(majorOp==31 && minorOp==983) ||    /* stfiwx   */    \
(majorOp==31 && minorOp==695) ||    /* stfsux   */    \
(majorOp==31 && minorOp==663) ||    /* stfsx    */    \
(majorOp==31 && minorOp==918) ||    /* sthbrx   */    \
(majorOp==31 && minorOp==439) ||    /* sthux    */    \
(majorOp==31 && minorOp==407) ||    /* sthx     */    \
(majorOp==31 && minorOp==725) ||    /* stswi    */    \
(majorOp==31 && minorOp==662) ||    /* stwbrx   */    \
(majorOp==31 && minorOp==150) ||    /* stwcx.   */    \
(majorOp==31 && minorOp==183) ||    /* stwux    */    \
(majorOp==31 && minorOp==151) ||    /* stwx     */    \
(majorOp==31 && minorOp==135) ||    /* stvebx   */    \
(majorOp==31 && minorOp==167) ||    /* stvehx   */    \
(majorOp==31 && minorOp==199) ||    /* stvewx   */    \
(majorOp==31 && minorOp==231) ||    /* stvx     */    \
(majorOp==31 && minorOp==487) ||    /* stvxl    */    \
(majorOp==58) ||  /* PPC64: ld, ldu, lwa  */    \
(majorOp==62) ||  /* PPC64: std, stdu     */    \
(majorOp==31 && minorOp==21) ||     /* PPC64: ldx     */    \
(majorOp==31 && minorOp==53) ||     /* PPC64: ldux    */    \
(majorOp==31 && minorOp==84) ||     /* PPC64: ldarx   */    \
(majorOp==31 && minorOp==341) ||    /* PPC64: lwax    */    \
(majorOp==31 && minorOp==373) ||    /* PPC64: lwaux   */    \
(majorOp==31 && minorOp==149) ||    /* PPC64: stdx    */    \
(majorOp==31 && minorOp==181) ||    /* PPC64: stdux   */    \
(majorOp==31 && minorOp==214)       /* PPC64: stdcx.  */    \
)

#define tt6_isMemoryExtended(majorOp, minorOp) ( \
(majorOp==31 && minorOp==533) ||    /* lswx - emit XER           */    \
(majorOp==31 && minorOp==661) ||    /* stswx - emit XER          */    \
(majorOp==31 && minorOp==342) ||    /* dst, dstt - emit rB       */    \
(majorOp==31 && minorOp==374)       /* dstst, dststt - emit rB   */    \
)
```

## TT6E Instruction Class Macros

```
#define tt6e_isFlowAltering(majorOp, minorOp) ( \
(majorOp==18) ||                    /* b, ba, bl, bla       */    \
(majorOp==16) ||                    /* bc, bca, bcl, bcla   */    \
(majorOp==19 && minorOp==528) ||    /* bcctr, bcctrl        */    \
(majorOp==19 && minorOp==16)  ||    /* bclr, bclrl          */    \
(majorOp==19 && minorOp==50)  ||    /* rfi                  */    \
(majorOp==17) ||                    /* sc                   */    \
(majorOp==19 && minorOp==18)        /* PPC64: rfid          */    \
)

#define tt6e_isMemory(majorOp, minorOp) ( \
(majorOp==34) ||  /* lbz      */          \
(majorOp==35) ||  /* lbzu     */          \
(majorOp==50) ||  /* lfd      */          \
(majorOp==51) ||  /* lfdu     */          \
(majorOp==48) ||  /* lfs      */          \
(majorOp==49) ||  /* lfsu     */          \
(majorOp==42) ||  /* lha      */          \
(majorOp==43) ||  /* lhau     */          \
(majorOp==40) ||  /* lhz      */          \
(majorOp==41) ||  /* lhzu     */          \
(majorOp==46) ||  /* lmw      */          \
(majorOp==32) ||  /* lwz      */          \
(majorOp==33) ||  /* lwzu     */          \
(majorOp==31 && minorOp==597) ||    /* lswi     */          \
(majorOp==31 && minorOp==119) ||    /* lbzux    */          \
(majorOp==31 && minorOp==87)  ||    /* lbzx     */          \
(majorOp==31 && minorOp==631) ||    /* lfdux    */          \
(majorOp==31 && minorOp==599) ||    /* lfdx     */          \
(majorOp==31 && minorOp==567) ||    /* lfsux    */          \
(majorOp==31 && minorOp==535) ||    /* lfsx     */          \
(majorOp==31 && minorOp==375) ||    /* lhaux    */          \
(majorOp==31 && minorOp==343) ||    /* lhax     */          \
(majorOp==31 && minorOp==790) ||    /* lhbrx    */          \
(majorOp==31 && minorOp==311) ||    /* lhzux    */          \
(majorOp==31 && minorOp==279) ||    /* lhzx     */          \
(majorOp==31 && minorOp==597) ||    /* lswi     */          \
(majorOp==31 && minorOp==20)  ||    /* lwarx    */          \
(majorOp==31 && minorOp==534) ||    /* lwbrx    */          \
(majorOp==31 && minorOp==55)  ||    /* lwzux    */          \
(majorOp==31 && minorOp==23)  ||    /* lwzx     */          \
(majorOp==31 && minorOp==7)   ||    /* lvebx    */          \
(majorOp==31 && minorOp==39)  ||    /* lvehx    */          \
(majorOp==31 && minorOp==71)  ||    /* lvewx    */          \
(majorOp==31 && minorOp==103) ||    /* lvx      */          \
(majorOp==31 && minorOp==359) ||    /* lvxl     */          \
(majorOp==31 && minorOp==310) ||    /* eciwx    */          \
(majorOp==31 && minorOp==438) ||    /* ecowx    */          \
(majorOp==38) ||  /* stb      */    \
(majorOp==39) ||  /* stbu     */    \
(majorOp==54) ||  /* stfd     */    \
(majorOp==55) ||  /* stfdu    */    \
(majorOp==52) ||  /* stfs     */    \
(majorOp==53) ||  /* stfsu    */    \
(majorOp==44) ||  /* sth      */    \
(majorOp==45) ||  /* sthu     */    \
(majorOp==47) ||  /* stmw     */    \
(majorOp==36) ||  /* stw      */    \
(majorOp==37) ||  /* stwu     */    \
```

```
(majorOp==31 && minorOp==247) ||    /* stbux    */    \
(majorOp==31 && minorOp==215) ||    /* stbx     */    \
(majorOp==31 && minorOp==759) ||    /* stfdux   */    \
(majorOp==31 && minorOp==727) ||    /* stfdx    */    \
(majorOp==31 && minorOp==983) ||    /* stfiwx   */    \
(majorOp==31 && minorOp==695) ||    /* stfsux   */    \
(majorOp==31 && minorOp==663) ||    /* stfsx    */    \
(majorOp==31 && minorOp==918) ||    /* sthbrx   */    \
(majorOp==31 && minorOp==439) ||    /* sthux    */    \
(majorOp==31 && minorOp==407) ||    /* sthx     */    \
(majorOp==31 && minorOp==725) ||    /* stswi    */    \
(majorOp==31 && minorOp==662) ||    /* stwbrx   */    \
(majorOp==31 && minorOp==150) ||    /* stwcx.   */    \
(majorOp==31 && minorOp==183) ||    /* stwux    */    \
(majorOp==31 && minorOp==151) ||    /* stwx     */    \
(majorOp==31 && minorOp==135) ||    /* stvebx   */    \
(majorOp==31 && minorOp==167) ||    /* stvehx   */    \
(majorOp==31 && minorOp==199) ||    /* stvewx   */    \
(majorOp==31 && minorOp==231) ||    /* stvx     */    \
(majorOp==31 && minorOp==487) ||    /* stvxl    */    \
(majorOp==58) ||  /* PPC64: ld, ldu, lwa  */    \
(majorOp==62) ||  /* PPC64: std, stdu     */    \
(majorOp==31 && minorOp==21) ||     /* PPC64: ldx     */    \
(majorOp==31 && minorOp==53) ||     /* PPC64: ldux    */    \
(majorOp==31 && minorOp==84) ||     /* PPC64: ldarx   */    \
(majorOp==31 && minorOp==341) ||    /* PPC64: lwax    */    \
(majorOp==31 && minorOp==373) ||    /* PPC64: lwaux   */    \
(majorOp==31 && minorOp==149) ||    /* PPC64: stdx    */    \
(majorOp==31 && minorOp==181) ||    /* PPC64: stdux   */    \
(majorOp==31 && minorOp==214) ||    /* PPC64: stdcx.  */    \
(majorOp==31 && minorOp==758) ||    /* dcba     */    \
(majorOp==31 && minorOp==86)  ||    /* dcbf     */    \
(majorOp==31 && minorOp==470) ||    /* dcbi     */    \
(majorOp==31 && minorOp==54)  ||    /* dcbst    */    \
(majorOp==31 && minorOp==278) ||    /* dcbt     */    \
(majorOp==31 && minorOp==246) ||    /* dcbtst   */    \
(majorOp==31 && minorOp==1014)||    /* dcbz     */    \
(majorOp==31 && minorOp==982)       /* icbi     */    \
)

#define tt6e_isMemoryExtended(majorOp, minorOp) ( \
(majorOp==31 && minorOp==533) ||    /* lswx - emit XER */         \
(majorOp==31 && minorOp==661) ||    /* stswx - emit XER */        \
(majorOp==31 && minorOp==342) ||    /* dst, dstt - emit rB */     \
(majorOp==31 && minorOp==374)       /* dstst, dststt - emit rB */ \
)
```